

LINUX VOICE

Inside the...

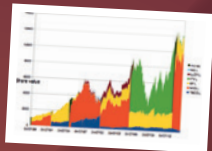
UBUNTU PHONE

The world's favourite Linux distro is coming to millions of pockets

SHARE DEALING

Get rich quick slowly

Write code to buy and sell shares, then weep as capitalism implodes



COMMUNITY

Start a FOSS project

You too can build a Free Software legacy – just make sure you do it right



38+ PAGES OF TUTORIALS



GROUP TEST Find the best distro for your NAS box

VECTOR GRAPHICS Adorn your website with SVG images

X86 Get your microscope and see how your computer really works

RASPBERRY PI

EBEN UPTON

"You could just sit on your arse and in two years time, everything will be twice as fast."



CRYPTOCURRENCIES

ALTCOINS

How cryptography, privacy and hard maths are shaking up the world of finance



FREE SOFTWARE | FREE SPEECH

May 2015 £5.99 Printed in the UK



ISSN 2054-3778

ANDREWS & ARNOLD LTD

will make you the

LINE KING



**BE THE MASTER OF
YOUR OWN TELEPHONE**

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

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

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

HAKUNA MATATA

Linux for human beings

The **May** 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 that 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 Michel Loubet-Jambert
michel@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:

Mark Crutch, Andrew Conway, Tim Elliot, Marco Fioretti, Josette Garcia, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Les Pounder, 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.

When you first turn on an Ubuntu Phone, you don't expect to see that familiar colour scheme, or the launch panel, or Unity. But all the old familiar elements that make up the Ubuntu desktop have made the transition. They're all there, running in the palm of your hand. It's not the desktop, of course. It's a completely different way of interacting with technology, but this is still a considerable achievement for a relatively small company. Canonical has been able to take an idea – immortalised by the Ubuntu Edge crowdfunding campaign – and turn it into reality.

What Canonical is doing is important. It's providing an alternative and giving us more choice. And while there are several open mobile alternatives, including Firefox OS, Tizen and even Android, none would be possible without Linux. Open source is the great enabler. It enables companies like Canonical and Mozilla to go their own way. It builds an ecosystem where choice and competition flourish, hopefully pushing us ever closer to technology that works for us, and not against us.

Graham Morrison
Editor, Linux Voice

**SUBSCRIBE
ON PAGE 64**



What's hot in LV#014



MAYANK SHARMA

"I've been thinking of buying into some cryptocurrency, so Ben's overview of what it is and why it's important is vital reading." **p28**



BEN EVERARD

"Hearing Eben Upton explain how Eric Schmidt caused him to return to Raspberry Pi HQ and cancel all model 2 R&D." **p44**



MIKE SAUNDERS

"Fortran passed me by the 1st time around (and the 2nd and 3rd), but it sounds ace so I'm going to try it now." **p100**



CONTENTS

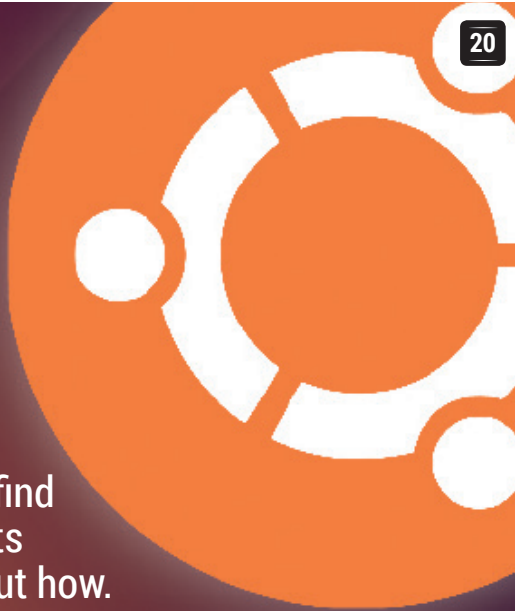
May LV014

Filled to bursting point with tutorials, tips and hacks, it's your new Linux Voice!

**SUBSCRIBE
ON PAGE 64**

UBUNTU PHONES

In 2015 Linux is going to find itself in millions of pockets around the world – find out how.

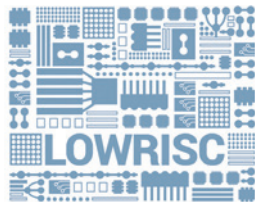


44 **Eben Upton**

We learn what it's like to drag the UK's education curriculum kicking and screaming into 2015.



28 ALTCOINS
Bitcoins, Litecoins, Darkcoins and more, explained and demystified for the curious.



36 OPEN HARDWARE
You have no way of knowing that your closed hardware isn't spying on you – that's why we need LowRISC.

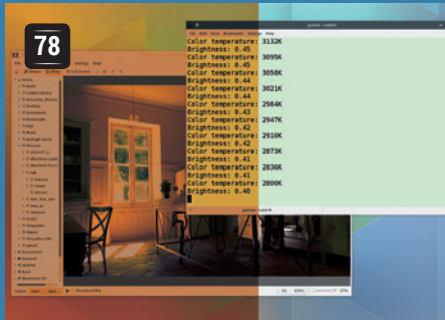


32 START A FREE SOFTWARE PROJECT
So you have a great idea for a FOSS project – here's what to do next.

REGULARS

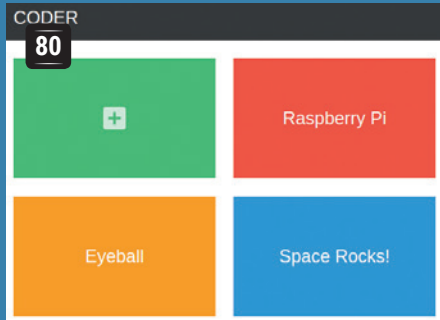
- 06 News**
GnuPG gets a much-needed cash boost, and Linux kernel developers get hired.
- 08 Distrohopper**
Find your next favourite Linux distro, which this month will be Tails, ArchBang or KaOS.
- 10 Gaming**
Zombies, cards and a game on a Grecian urn vie for our in-demand attention.
- 12 Speak your brains**
Share your thoughts – there are no prizes for doing so, but we might thoughtfully nod.
- 16 LV on tour**
Our agents report from Jersey, Ipswich and the London PostgreSQL meetup.
- 42 FAQ: HTTP/2**
Like our Victorian sewers, the internet's plumbing desperately needs an upgrade.
- 58 Group test**
Find the best operating system for your network attached storage.
- 64 Subscribe!**
Save money on your monthly fix of Linux Voice by getting it delivered to your door/inbox.
- 66 Sysadmin**
Meet every system administrator's best friend – the Webmin interface.
- 68 FOSSpicks**
Try the best Free Software in existence – it's all there on the other end of a download.
- 110 Masterclass**
Keep a close eye on your network traffic with Wireshark and Tshark.
- 114 My Linux desktop**
Enter the geek den of Selene Scriven, professional Ubuntu breaker.
- 18 FOSDEM**
Mike and Ben visit Brussels to celebrate the biggest Free Software gathering in Europe.

TUTORIALS



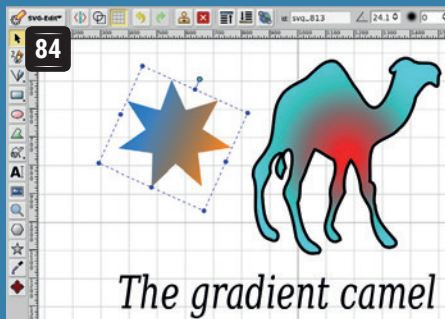
78 Redshift: Ease eye strain and sleep easier

Reset your body clock with free software and the Kelvin scale.



80 HTML, CSS and JavaScript on the Pi

Learn web essentials the easy way with Google Coder.



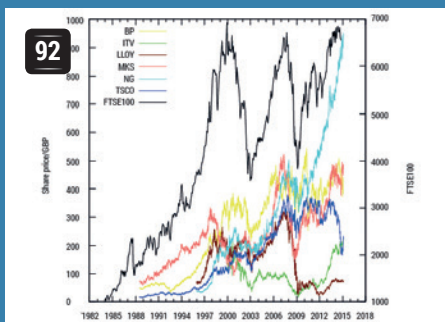
84 Vector graphics on the web, for the web

Prettify your web pages with low-bandwidth SVG images.



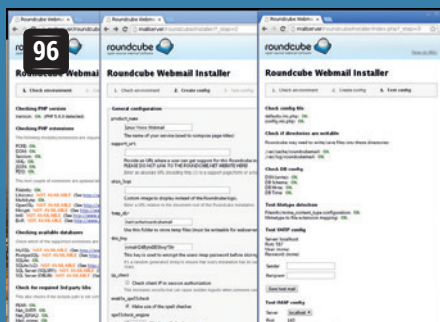
88 How your computer works: inside an X86 chip

Get down to the nuts and bolts of what your machine is doing.



92 Stockmarket analysis with open code

A few lines of Java are all it takes to beat the FTSE 100 index...



96 Roundcube and Cyrus: Set up webmail

Keep Google's prying eyes out of your email communication.

100 Fortran: Coding for scientists

Code from IBM's 1950s glory days.

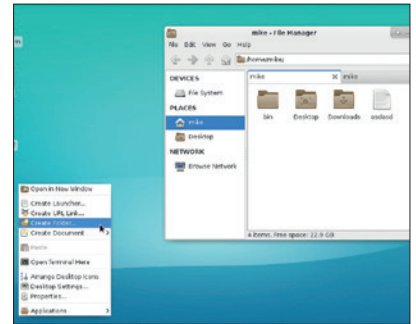
104 Code ninja: Code reuse

Recycle code to save time and effort.

106 Assembler: Code on bare metal

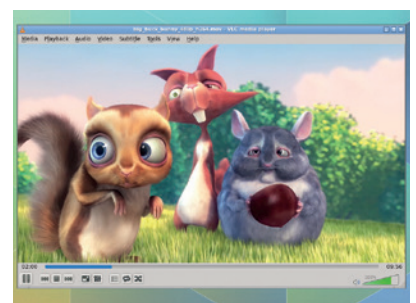
Who needs an operating system?

REVIEWS



50 Xfce 4.12

The latest desktop from the Xfce team has the apps, the looks and the usability



52 VLC

The Swiss Army knife of multimedia playback adds another layer of polish.

53 Krita 2.9

Move over Gimp – you've just lost your claim to be the best image editor on Linux.

54 Inkscape 0.91

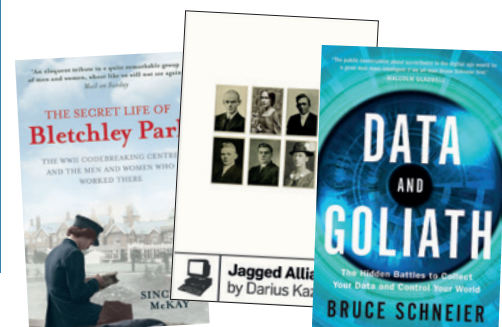
The *numero uno* vector graphics software gets its first big update in four years. Four years!

55 IPython 3.0

An interactive shell for analysing data sets, based on the Python language.

56 Books

Bletchley revisited, cult gaming and the latest from security guru Bruce Schneier.



NEWS ANALYSIS

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

Opinion

Certifications and jobs: do they go together?

If you're the one selling the certificates, of course they do. If you're looking for a job, maybe not...



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

It's spring and change is in the air – not least with yours truly, who has started a new job at global consulting firm Wipro. The market for open source related jobs seems strong at the moment. In a report published by the Linux Foundation and built with help from jobs website www.Dice.com, demand for Linux professionals and those with cloud, security and networking expertise showed up as stronger than ever. They surveyed around 1,000 hiring managers and 3,000 Linux professionals and found that almost all the hiring managers have multiple vacancies for Linux and open source skills.

That's really no surprise if you've been watching the state of technology adoption. Open source is now the assumed default at every level of every new business solution. So when the Linux Jobs Report highlights open source cloud computing skills as a priority across those surveyed – that's open source cloud – 42% of hiring managers are seeking OpenStack and CloudStack experience in their candidates.

So how do you get those jobs? The Linux Foundation wants us to believe that the training and certification they sell is the key,

just like proprietary vendors before them. Those with certification products to sell naturally want us to think that's the way the world works. But I'm hugely sceptical of such things.

Tests don't have to be testing

Tests like the ones from LPI (and sadly a large number of commercial vendors too) tend to be multiple choice tests that favour memorisation rather than experience. One experienced professional I asked scoffed at the very thought of multiple choice tests as an indicator of Linux competence, telling me that these tests, including pretty much all of IBM's, don't reward real-world skills. They reward memorisation, not the ability to work with the technology, not problem solving, not even the ability to find answers; just the ability to have memorised certain facts.

Certifications that test actual skills – like the ones from Red Hat and The Linux Foundation – were rated more highly. One person told me “I contrast [those certifications] starkly with an exam like Red Hat's, which is a practical, hands-on, test.”

If professionals have such a low opinion of certifications, why are they so common? One professional told me:

“They should be seen as naught but a crutch for HR people, and are an extremely poor filter.”

Certifications are a tool used by recruiters to thin the pack of applications for a vacancy and hopefully identify the one worthy of further scrutiny. The actual hiring manager is much less likely to be interested; she will want proof of hands-on experience. If

that's not available, a track record from an open source community – preferably as a committer – is next best.

Sadly, respect for certifications has spread more widely; it's used by some governments as a bulwark against system failures. There are already 10 states in the United States, as well as all of Canada, that regulate the use of the term “software engineer” and seek a professional certification from those working on high-profile systems.

That's unlikely to help make systems safer though. In most cases, projects fail because managers pick proprietary technologies from suppliers based on a range of criteria that have little bearing on the suitability of the software from the perspective of a software engineer. Rather, choices are influenced by corporate marketing, by existing contractual relationships, by technology lock-in, and sometimes even by relationships with senior executives. Enterprise middleware often gets picked on the golf course.

Experience always wins

Certifications for developers don't guarantee project success. That's guaranteed more by process transparency, open source software, and system audits. If skilled engineers select technologies purely on merit and go on to defend their choices under audit, they're likely to prefer open solutions, which they are free to scrutinise independent of vendor oversight.

For me, getting hired by Wipro was not a function of any certifications – indeed, the last qualification I sought was in 1982. The company approached me based on my reputation and proven work history, as well as my community engagements. In a “seller's market”, that's surely how the majority of jobs will be filled?

“42% of hiring managers are seeking OpenStack and CloudStack experience in their candidates.”

CATCHUP

Summarised: the biggest news stories from the last month

1 Next Linux kernel version to be 4.0, not 3.20

Linus Torvalds has never been a fan of big kernel version numbers. So he has decided that the next release will be 4.0, rather than 3.20 as expected, and there will be plenty of goodies for us to explore as well. Most significantly, live kernel patching support will be included, making it possible to update your kernel without having to reboot – a huge win for servers where uptime is critical. Kernel 4.0 will also support more ARM system-on-chip devices, and IBM z13 mainframes.

2 CrunchBang Linux dies, comes back to life

CrunchBang, the minimalist Debian-based distro with *Openbox* as its window manager, is no more. Lead developer Philip Newborough explained that CrunchBang was fun, but the distro landscape has changed so much in recent years and it “no longer holds any value”. This was sad news for many fans, but it didn't take long before the community created a successor. It's early days, but a release candidate should be available soon:

www.crunchbangplusplus.org

3 Xfce 4.12 released, after three years development

It's been a long time coming, but lightweight *GTK*-based desktop *Xfce* has a new release with features and fixes aplenty. See our reviews section on page 49 for the full lowdown.

www.xfce.org



4 Huge fundraising drive gives GnuPG a future

GnuPG is one of the most widely used tools for email encryption, but in early February its only main developer, Werner Koch, lamented that he couldn't afford to keep working on the project. Times looked bad, but a massive fundraising effort by the community has given *GnuPG* a very healthy future: over \$150,000 was donated from individuals and companies, while Facebook and Stripe have pledged another \$50,000 on top.

www.gnupg.org

5 Kernel developer sues VMware over GPL

Christoph Hellwig, one of the 20 most active Linux Kernel developers, is suing VMware, makers of a virtualisation product. Hellwig alleges that the firm is illegally using GPL code he's written in part of their proprietary software. The Software Freedom Conservancy (SFC) is funding the case in Hamburg, Germany. The SFC has been in discussions with VMware since 2007 over compliance, but this is the first time it's reached court.

<http://tinyurl.com/qf3fc4b>

6 Linus Torvalds: “Kernel coders get hired quickly”

Fancy making megabucks as a software developer? Hacking on the kernel is possibly the best way to land a lucrative job – at least, according to lead developer Linus Torvalds. He was responding to new statistics showing that just 12% of kernel code in the last year was written by non-paid volunteers, down from 19% the year before. Torvalds' reasoning: it's not that there are fewer volunteers, but those who prove themselves with good patches end up getting jobs quickly.

7 Elementary OS seeks cash, generates hate

Ubuntu spin-off Elementary has a “suggested donation” part of its download page. Recently this was \$0, but the developers bumped it up to \$10 with this reasoning: “We want users to understand that they're pretty much cheating the system when they choose not to pay for software”. Uh oh. This caused uproar in the Elementary community, and the wording has since been changed, but not before everyone pointed out that Elementary should give money to Ubuntu (and Debian).



8 Thunderbird doing well, will get a calendar

While the Mozilla Foundation puts most of its efforts into the *Firefox* web browser and *Firefox OS* mobile platform, good old *Thunderbird*, the mail client, hasn't been forgotten. Despite a relatively slow development pace, the download rate for *Thunderbird* is increasing each month, and version 38 will be released very soon with a calendar. This will be provided by the *Lightning* add-on, and bring *Thunderbird* up to par with *Microsoft Outlook*.

<http://tinyurl.com/puzoom7>

DISTROHOPPER

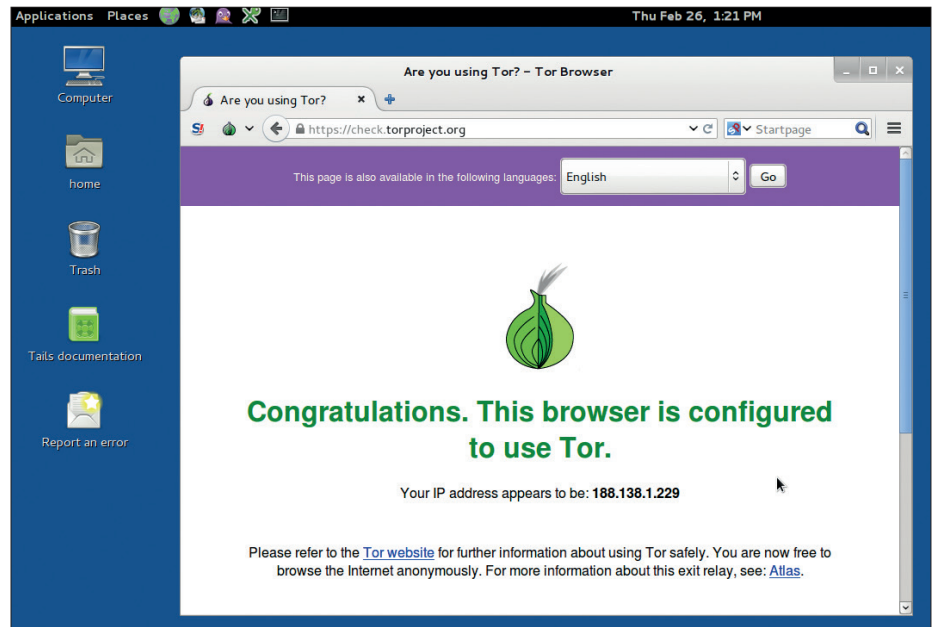
Our pick of the latest releases will whet your appetite for new Linux distributions.

Tails 1.3

Secret and safe.

Governments and law enforcement agencies may be hell-bent on monitoring every single thing we do and say, but geeks are fighting back. Tails (<https://tails.boum.org>) is a live Linux distribution that routes all internet traffic through the Tor anonymising network, and it doesn't leave any traces on your hard drive. Why is this important? Well, imagine you're a journalist working in a repressive regime. With Tails, you can boot the distro from a DVD or USB key, do your work online, reboot and destroy the Tails media. If your machine is confiscated, there's no way to tell that you were using Tor – at least, that's the goal.

The Tails developers are keen to point out, however, that it's not a 100% perfect solution for anonymity. Network traffic is encrypted as it moves around between Tor nodes, but it's plain to see when it leaves an exit node, and you can't guarantee that all exit nodes are operated by do-gooders. Also, you have to be very careful when signing into services on the web. There's no point using Tails and



Tails 1.3 includes a new Bitcoin wallet and the obfs4 pluggable transport to disguise Tor traffic.

or to stay anonymous if you then sign into Facebook or other data-harvesting services.

Tails 1.3 boots to a Gnome desktop with *Firefox*, *Claws Mail* and the *Pidgin* instant messenger, all using the Tor network. If any program tries to access the internet directly,

and not through Tor, it gets blocked. Plus there are some extra tweaks for privacy – like the use of StartPage as the default search engine in *Firefox*, which uses Google to get search engine results, but doesn't share your IP address.

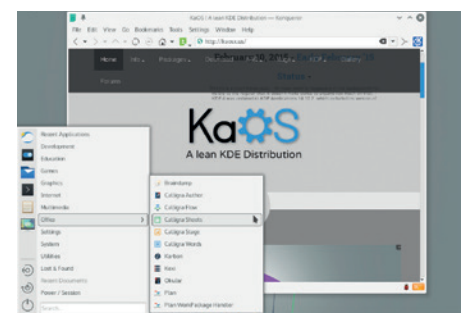
KaOS 2015.02

Gorgeous rolling distro showcasing the latest KDE.

KaOS (<http://kaosx.us>) is described on its website as a “lean KDE distribution” – but isn't that something of an oxymoron? KDE is the most featureful and customisable desktop environment in existence, but few would describe it as “lean”. In KaOS, though, that word is used to describe the whole experience and not just the desktop. KaOS focuses entirely on one desktop (KDE), one toolkit (*Qt*), one architecture (x86-64) and one release model. Unlike many distros, it doesn't try to be a jack of all trades; it has a razor-sharp focus, and won't budge from it.

Well, for now at least. The KaOS developers are considering a switch away from the Linux kernel in the future, possibly to Illumos (a derivative of OpenSolaris). We're always open to other FOSS operating systems at Linux Voice, but we're not sure what Illumos would bring – sure, it has some high-end features useful if you're running giant databases on big iron, but it doesn't have anywhere near the same level of x86 hardware support.

KaOS 2015.02 uses the KDE Plasma 5 desktop, and is one of the best-looking KDE configurations we've ever seen. Subtle



KaOS is a rolling-release distro, so you install it once and receive gradual updates.

animations and drop shadows abound, while the fonts and icons complete the package excellently. You won't find many non-KDE or *Qt* programs here – *Konqueror* is the only web browser installed by default, and *Calligra* provides the office suite.

ArchBang

Arch Linux goodness, without the installation hassles.

Back in issue 7's cover feature, we declared Arch Linux as the best all-round distribution. It's always up to date, it has a giant range of software via the Arch User Repositories, and its documentation is second to none. It's not ideal for newbies or for long-term Debian-esque stability, but for those who don't mind dabbling at the command line now and then, performing the occasional fix, it's bliss.

The installation process teaches you a lot about the underpinnings of a Linux system, but it can become a chore when you've done it several times. So ArchBang (www.archbang.org) has become one of our favourite distros in the last 12 months – it gets you up and running with an Arch installation quickly, so you can avoid the tedious parts of the installation process.

ArchBang is inspired by CrunchBang Linux, sporting a minimalist desktop with Openbox as the window manager and a small selection of tools. Unlike some Arch-based distros, ArchBang doesn't have its own special repositories – it's simply a means to get a working Arch setup in just a



ArchBang's default theme is dark and moody, but it's easy to brighten it up.

few keypresses. Its main feature is its installer, which simplifies the process of getting Arch onto your hard drive; it's a menu-driven tool and easy to navigate for intermediate and experienced Linux users.

Otherwise, the distro is pretty bare: it includes *Firefox* and a few other tools, but the idea is that you get it installed and then add what you need via *Pacman* and the regular Arch repositories.

MuLinux – desktop Linux on two floppy disks!

Here's a question: how much space do you need for a graphical Linux installation with a file manager, terminal emulator and various other utilities? If a typical Ubuntu installation weighs in at around 2.5GB, you might think that you could squeeze the bare necessities into 200MB or so. Well, go even further: it's actually possible to get some GUI goodness in under 3MB. MuLinux (<http://micheleandreaoli.org/public/Software/mulinux/>) is a long-defunct project that caught some attention in the early 2000s by managing to pack an extraordinary amount of Linux onto a few floppy disks.

The first disk provided the base system, and then you could beef it up with extra disks containing the X Window System, *GCC*, Perl, *Wine* and even a Java virtual machine (*Kaffe*). Most of these programs were highly stripped-down in order to fit into the limited space, but it was nonetheless an impressive achievement, and showed what's possible when you take every kilobyte into consideration. And today, it makes you wonder what on earth happened to modern operating systems to bloat them up so much – why does Windows 10 need 16GB of hard drive space?

Although MuLinux ceased development over a decade ago, there have been similar projects in recent years. Tiny Core Linux (<http://distro.ibiblio.org/tinycorelinux>) crams a GUI Linux distro into just 12MB, and while it's not all that useful on its own, you can use it as a base for bigger projects such as a web kiosk or similar installation.

It may look pants today, but MuLinux was an astonishing feat at the time.



GAMING ON LINUX

The tastiest brain candy to relax those tired neurons



WAITING FOR THE SUN



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

The Linux gaming world currently has all eyes on the Game Developer Conference this March, as what happens there is likely to set the precedent for the future, with 2015 looking to be the most eventful year for Linux gaming to date.

Valve will have centre stage at the Conference to showcase its PC-console hybrid Steam Machines, with which they intend to take a chunk of the market. It's unlikely to be an official launch, but we can expect something of great importance, be it revealing more technical details or coaxing developers into putting games out on Linux, specifically the company's Debian-based SteamOS.

Secondly, OpenGL's successor GLNext will be officially revealed there. Not much is yet known about GLNext other than it being a big initiative with a lot of industry names behind it, so the announcement is keeping many on the edge of their seats. This transition is long overdue and needed to get more developers porting to Linux or even ditching DirectX (also seeing its next big release this year) in favour of GLNext.

Many developers have ported their games to Linux with the prospect of Steam Machines being a success, and may not continue if they're a flop, while the unmanageability of OpenGL has been their major complaint and stumbling block. There is a lot riding on both, and it's no coincidence that both announcements are happening under the same roof.

Dying Light

Not just your standard zombie survival game.

Zombie games are seemingly a dime a dozen nowadays, however *Dying Light* stands out from the horde of rotting flesh, bringing a lot of new stuff to the table.

The game's unique selling points are its parkour and day/night mechanics. The parkour elements allow for more use of space while giving more options than simply taking on the horde directly, while nighttime gameplay gives the game more of a survival-horror feel than an open-world sandbox, as zombie behaviour becomes more aggressive by night, summed up best by the game's "hunter by day, hunted by night" tagline. Combined with features like crafting and a narrative that goes beyond the standard fetch quests interspersed with the occasional cutscene, the game reinvigorates an otherwise stale genre.

Speaking of which, it's almost impossible not to make comparisons to *Dead Island*, which shares the same developers. Those familiar with *Dead Island* (ported to Linux last year) will quickly find many similarities between the games, with certain mechanics and even the designs of the



With these next-gen graphics, some of us may have to think about upgrading graphics cards. POW!

weapons looking like they were ripped out of *Dead Island* and shoved into *Dying Light*.

That said, *Dying Light* is still somewhat of a cautious recommend for the time being. There are plenty of bugs, and unless you have the latest and greatest graphics card, you shouldn't expect good frame rates. The game is tonnes of fun, but those wanting a smoother experience should hold back until more patches come out.

Website <http://store.steampowered.com/app/239140> Price £39.99



A couple of patches have already rolled out and performance is due to get better. OOF!

"Nighttime gameplay gives Dying Light more of a survival-horror feel than an open world sandbox."

The Book Of Unwritten Tales 2

Adventure games are far from dead.

If you're a fan of the adventure genre, you shouldn't think twice about getting this one. *The Book of Unwritten Tales 2* hits the mark on every level, from the lovable characters, well thought-out puzzles and beautiful settings to its writing and humour.

Unlike the previous game, this features an array of entertaining side-quests and optional extras which add a replay value often lacking in a normally linear genre. It's details like these that make this game feel more like a labour of love rather than a product put out by a successful studio. Something else the game does

exceptionally well is pulling off nostalgia and game references. This game certainly meets the standards set by the likes of the *Monkey Island* franchise and exceeds them in many ways, being easily one of the best written, funniest and best looking adventure games to date.

As it should be clear from the title, the game is a sequel and although it stands up extremely well on its own, playing the previous instalment does enhance the experience greatly.

Website <http://store.steampowered.com/app/279940> Price £29.99



King Art seems intent on perfecting the adventure genre.

Hand of Fate

Deckbuilding has never been so enthralling.

Mixing together different genres tends to be a recipe for disaster, however, *Hand of Fate* has done so masterfully. The main setbacks of a card game are repetitiveness and randomness, which this game has mitigated by introducing RPG elements and combat to make victory more dependent on player skill. All this comes together to create a unique experience.

Casual is a term with negative connotations in gaming today. While *Hand of Fate* is a game that allows sporadic playing for those too busy to sink continuous hours into a game, it can also be very captivating, so perhaps smart-casual would be a better suited category. Casual also doesn't necessarily mean easy, with poor deck choice and sloppy combat prematurely ending games and the *Rogue*-like elements often being



Have you got what it takes to play the game?

punishing. It should be clear by now that this is far more than a deckbuilding game, and an underlying story develops as the player progresses through the game. Just who is this mysterious dealer? Is he friend or foe? Who is the protagonist and why is he playing this game? These elements add to the already addictive gameplay.

Website <http://store.steampowered.com/app/266510> Price £18.99

ALSO RELEASED...



Grim Fandango Remastered

Often considered one of the greatest adventure games of all time, *Grim Fandango* has had a much needed lick of paint from Tim Schafer's Double Fine Productions which also made this masterpiece officially available on Linux for the first time. Some of the new features include a more classic point-and-click interface, improved lighting and tweaked musical score, breathing new life into the game's characters and rich film noir world. <http://store.steampowered.com/app/316790>



Fahrenheit: Indigo Prophecy Remastered

Another remaster, this time brought to us by Aspyr Media, which ported *Civilisation V* and *Borderlands 2* to Linux. Originally released in 2005, *Fahrenheit* sees you uncover the mysterious goings on surrounding a series of murders taking place in New York City, using innovative gameplay still ahead of its time. <http://store.steampowered.com/app/312840>



Apotheon

Aside from being stunningly beautiful, this game is also an incredibly solid *Metroidvania*-style platformer with a good dose of story. *Apotheon's* biggest strength is using its combination of ancient Greek style and ominous music to make the player feel like they're a hero in a Greek epic. You can tell the devs have put a lot of work in researching the period, its mythology and literature. <http://store.steampowered.com/app/208750>

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

MIKE WINS

Just a quick note to say hello. I've just subscribed after buying issue 12 and am enjoying 1–4 as back issues too.

It's the assembly language tutorial that did it. Low-level knowledge will be relevant as long as we need hardware to run things on; even the cloud runs on hardware (eventually). This took me back to learning Z80 as a spotty kid on my old Amstrad CPC.

And now I'm doing the kernel module tutorial from issue 2. Brilliant stuff, I'll subscribe

as long as you have this kind of content in every issue.

Scott

Mike says: Sir, I am humbled. We're glad you like the magazine, and hope to provide you with much more good stuff in the months and years to come.

A long, long time ago, we published an image of a dartboard with Mike's face on it. It was wrong of us to do so, because he writes such lovely assembler tutorials. Here he is up against a wall instead.

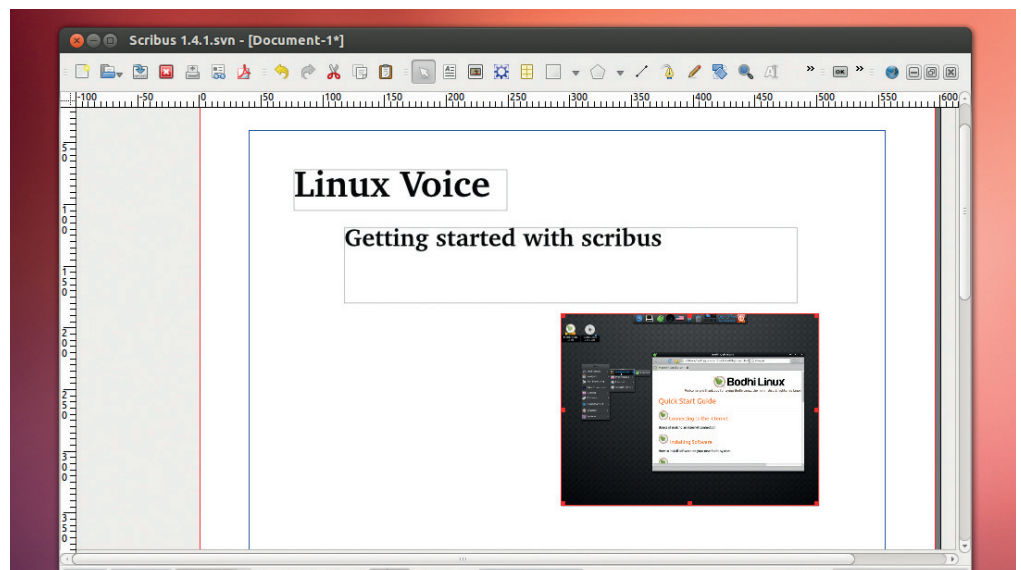


SCRIBUS

Now, come on chaps, where is that tutorial on *Scribus* you've been promising since the December issue? Here I am, one of your new fans, and what do I get? Disappointment, that's what! So come on, buck up and get it in the next issue, please!

A Geoffrey Mort

Andrew says: The fact that the *Scribus* feature is taking so long is down to the key problem with it that we've found so far. We want to test it out properly by putting it in the hands of our art boss, Stacey. She's intimately skilled in the way of Adobe *InDesign*, so is the perfect guinea pig for *Scribus* (an application pitching itself as a professional design tool ought to be usable by design professionals). Unfortunately, so far it's missing



some features we need. I'd very much like us to be able to support *Scribus* development – it's fantastic software

– so vote for it in our profit sharing scheme at www.linuxvoice.com/profitvote1 (the code is LV3276XJA).

Scribus isn't working for us, so it's our responsibility to provide useful, feedback.

MIND YOUR MANNERS

I was sad to read that Chris Brown is retiring and will no longer be writing for your magazine. I'd like to thank Chris for the enjoyment and learning I have derived from his articles over many years, both in LV and in "another" magazine. I have always found them to be lucid and pitched at the right technical level. Even when I read an article on a subject that I was familiar with, I knew that I would always learn something new.

I was interested to read, in "What the big names say" [in issue 12], that "Civility" was cited explicitly by Philip Newborough and implicitly by Boudewijn Rempt as a significant challenge for GNU/Linux and FOSS, especially as it's echoed elsewhere in the same issue as part of the interview with Lennart Poettering. On the one hand, it's good that people feel strongly on an issue and are prepared to argue a technical case and defend it (even to the extent of forking a code base and doing something new). On the other hand, personal attacks diminish the attacker in addition to doing the whole community a disservice. Sad to say, Linus himself is not blameless in this regard.

Of course, there are arguments and bad behaviour within commercial companies as well – including covert and overt



behaviour as bad as anything we see in the FOSS world. In general, commercial companies have the luxury of not airing their dirty washing in public. Maybe, sometimes, there are disadvantages to being open!

Thanks for a great magazine.

Neal Crook, Berkshire

Andrew says: That's something I'd never thought of until now. Name calling, shouting, abuse and bullying all go on in big companies as hierarchies are established and maintained, and we've all moaned about idiot bosses who we wouldn't trust to look after a goldfish, never mind a team of people. But this all goes on behind closed doors. Maybe we're not such a bad bunch of people after all!

When Linus shouts at developers, he does so in public. When Steve Ballmer used to shout at developers, he did so in private. Which would you prefer to be on the end of?

SO FAREWELL THEN

I loved reading Chris Brown's technical run down articles. I am very sad to see he will be retiring! Tell him he can't quit yet. I just renewed my yearly subscription!

Zane Williamson

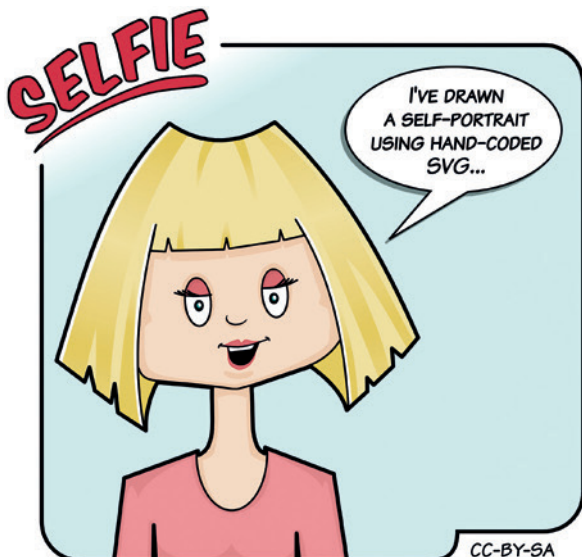
Graham says: Sorry, but Chris has this pesky thing called free will. I'd ban it if I could, but the next best thing is to persuade another Linux expert (who has a PhD in physics, like Chris) to take his place and write the system administration pages as of next issue: Dr Valentine Sinitsyn!

CRUNCHBANG

On the event of its passing into Linux history, I'd like to give a vote of thanks for the Crunchbang Linux distribution. It breathed new life into a couple of old machines, saving me a few quid; it saved me time, with its speedy boot routine; and it introduced me to a fantastic community of users who all just want to be nice to each other. Cheers Philip!

Gary Waterman, Braintree

Andrew says: Philip Newborough, chief bodger of Crunchbang, has had enough and won't be carrying on with the distro. But cry no more – there's very likely to be a community fork in the works...



LINUX ON THE HIGH STREET

The HP850 with SUSE Linux Enterprise at \$1,709 and the Librem 15 (starts at \$1,899) are aimed at specialist Linux users with plenty of money (to return to a theme from LV012 Letters).

What is really needed is a cheap laptop with a reasonable CPU, preferably running Linux Mint and aimed squarely at the general public.

Mint, with its expanding menus, looks very similar to Windows XP, Vista and W7, and it is a comfortable transition for those previously using those systems. In contrast, Ubuntu Unity follows the W8.1 path which just makes things unnecessarily difficult for new adopters.

Mint comes with virtually all the software that most users need and programs like Skype can be added easily.

To help digitally-excluded and “W8-baffled” locals who want new machines, I wipe W8.1 and install LM17.1 Mate on the HP 255 G3, which uses an AMD A4-5000 CPU (1905 benchmark). Wherever possible I install Mint on their existing machines.

The price of the HP 255 G3 fluctuates between £200 and £220: www.ebuyer.com/669147-hp-255-g3-quad-core-laptop-k7h92es-abu.

Go on, LV readers, do your bit and help someone you know get to grips with a Linux Mint computer.
Godfrey Green, Cardigan

Andrew says: Thanks Godfrey. It’s encouraging that you agree with Tony Hughes, who wrote in last issue’s letters pages about his experience of refurbishing old computers with Linux Mint for those who have been left

behind by Microsoft’s baffling user interface choices. It really does stand out from the ‘by geeks, for geeks’ tradition – whisper it, but it could even be said to be Linux for human beings...

The HP 225 G3 is an admirable, affordable workhorse of a laptop that’ll run Linux Mint no bother.



HEY, TEACHER!

I noticed your plan to send out free copies of LV to schools in the UK. A good move. So I emailed our local high school (I am in Sydney) to alert them about the free downloads available on your site. No reply yet but you never know.

If anyone else thinks their local educational establishment could do with a bit of free software goodness, why not give them a nudge and see whether they respond?

Tim Lloyd

Mike says: Please do; that’s what they’re there for. The UK (and Estonia, as we learn from Eben Upton on page 44) is leading the way in computing education, and we’d like to help that along by sharing our old issues. There’s a growing archive of content available to download for free at www.linuxvoice.com/creative-commons-issues, and the best bit is that it’s all freely licensed, so you can share it with as many people as you like, copy it, update it and use it however you see fit. Even if you’re in Australia!



Thinking about subscribing to Linux Voice? Issues 1–4 and more are available right now to download completely for free from www.linuxvoice.com/creative-commons-issues.

ENCRYPTION

I read with interest the article in the March issue from the FSFE [Free Software Foundation Europe] regarding email encryption.

Under the section headed "Practical Advice" the article states "Thus, the more you encrypt your messages, the less suspicious encrypted messages will be."

I am just taking my first faltering steps in the FOSS community, but surely implying that encryption might reasonably be regarded as doubtful or of nefarious intent cedes too much ground to those who would argue "...if you have nothing to hide, you have no need to encrypt your communications," and undermines the case for a default position where the right is to privacy.

When I post a letter I seal the envelope, not because it contains details of my next daring jewel heist, but because I value my privacy (as a key feature of a free and fair society), and anticipate the carrier will do the same and treat it with respect and discretion.

A FEW WORDS FROM THE FSFE: EMAIL ENCRYPTION

UNPROTECTED COMMUNICATION

Mass surveillance violates our fundamental rights and is a menace to the freedom of speech! But: we can defend ourselves.

The password that protects your email is not sufficient to protect your mails against the mass surveillance technologies used by secret services. Each email sent over the internet passes through many computer systems on the way to its destination. Secret services and surveillance agencies take advantage of this to read millions and millions of emails each day. Even if you think you have nothing to hide: Everyone else that you communicate with via connected emails is being exposed as well. Take back your privacy by using GnuPG! It encrypts your emails before they are sent, so only the recipients of your choice can read them.

GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available for change.

About GnuPG

Thousands of people already use GnuPG, for professional and private use. Come and join us! Each person makes our community stronger and proves that we are ready to fight back. Whenever an email that is encrypted with GnuPG is intercepted or ends up in the wrong hands, it is useless. Without the appropriate private key it cannot be read by anyone. But, for the interested in-

...and only for her - it opens like a totally normal email. Sender and recipient are both safe now. Even if some of your emails contain no private information, consistent use of encryption protects us all from unqualified mass surveillance.

What makes GnuPG secure?

GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control. If no one is allowed to see the source code of a program, nobody can be sure that it does not contain undesirable spy programs - so-called "backdoors". If software does not reveal how it works, we can never trust it blindly.

In contrast, a fundamental condition of Free Software is to publish its source code. Free Software allows and supports independent checking and public review of the applied source code by everyone. Given this transparency, backdoors can be detected and removed. Most Free Software lives in the hands of a community that works together to build secure software for everyone, if you want to protect yourself from surveillance you can only trust Free Software.

Practical advice

The technology behind GnuPG provides first-class protection. To ensure that your encrypted communication is not compromised for other reasons, use a strong passphrase and backup your private key. Encrypt as much as you can! By doing so, you prevent others from relaying when and with whom you exchange sensitive information. Thus, the more often you encrypt your messages, the less suspicious encrypted messages will be. Be aware that the subject is transmitted unencrypted.

Watch out for so-called "Oxytocin parties" in your area! These are events where you can meet people that would be happy to help you in setting up and using GnuPG as well as other encryption tools at no charge. fsfe.org/en

About the fsfe

This article was created by the Free Software Foundation Europe (FSFE), a non-profit organisation dedicated to empower people in Europe in their use of technology by promoting software freedom.

What is Free Software?

Free Software can be used by everyone for any purpose. That includes free copying, reading the source code and the possibility to improve or adapt it to your own needs (the so-called "four freedoms").

Even if you "only want to use" the program, you still benefit from these freedoms. Because they guarantee that Free Software remains in the hands of our society and that its further development is not controlled by the interests of private companies or governments.

Find out more about this and how Free Software can lead us into a better future. fsfe.org/freesoftware

If you like to help spreading the word, you can order this and other leaflets under: fsfe.org/leaflets

Donations are official for us to continue our work and to guarantee our independence. You can support our work best by becoming a sustaining member of the FSFE, a "fellowship". By doing so, you directly help us to continue the fight for Free Software wherever needed. fsfe.org/en

You can find a simple tutorial for email self-defence with GnuPG encryption here: EmailSelfDefense.FSF.org

fellowship of free

Surely this must be our starting point and not a tacit concession that the right to privacy must be justified.

Steve Brodie, Thetford

Andrew says: We'd love to take credit for those pages, but they were provided by the FSFE. Not because

the FSFE gave us money, but because we believe in the message and want to share it: encrypted emails make everyone better off, even if they're about something as banal as your shopping list or plans for the weekend. You're right that privacy should be accepted by default, but unfortunately it isn't.

Unencrypted email is as secure as sending a postcard - ie not very.



**YOUR AD
HERE**

LUGS ON TOUR

Open Source Day, Jersey

We were invited to attend one of the Channel Islands' inaugural open source event.

We learnt lots of things when we visited Saint Helier, the capital dwelling on Jersey, an island in the English channel a mere 12 nautical miles from Normandy, France. We learnt that legal text is written in a French dialect, Victor Hugo spent considerable time here while exiled from France and that you can't spend money given to you as change back when you get back to England. Jersey is one of those anachronisms from British history – a Crown Dependency with the right to self governance.

Open source is enthusiastically used everywhere by many geeks on Jersey, but there are still areas of the local business community that need some convincing. Technology on Jersey seems to exist within its own microclimate.

This is why Matt Chatterley and a group of local developers and hackers decided to organise Jersey's first 'Open Source Day'. Despite the potentially limiting size of the local population (around 100,000 for the entire island), the event itself was a considerable success, introducing lots of new people to many of the ideas we all take for granted. As well as an anecdotal talk given by our own Graham Morrison, a speech which served as the keynote and included images of an Acorn Electron and a Commodore 64, there was a single track of workshops throughout the day. Tom Brossman kicked off the event by talking about A+ SSL with Ubuntu, followed up by a brilliant talk about exploiting buffer overflows by Paul Dutot. Paul impressively did this for real, demonstrating an exploit to take

#OPENJSY
11.02.15
DIGITAL JERSEY & COLLABORATE.JE PRESENT

OPEN SOURCE DAY

• EMBRACE OPEN SOURCE IN 2015 •

<p>Wed 11th February 2015 10:00am - 4:00pm Digital Jersey Hub Forum 3 - Grenville Street St Helier - Jersey Register on Eventbrite</p>	<p>The Trade Show will include:</p> <ul style="list-style-type: none"> • Keynote Speaker (12-2pm) • Workshops • Exhibitions • Interactive Stands • Lunch Provided 	<p>@collabje @digitaljersey @escapeint</p> <p>www.collaborate.je www.digital.je www.e-scape.co.uk</p>
--	--	---

Sponsored by **e-scape**

Jersey's traditional business is finance and banking – two areas ripe with open source potential.

control of a Windows PC in front of his audience.

In the afternoon, Rob Dudley gave a great beginner's overview of *WordPress*, which was followed at the end of the day by a talk by Jason Stratford on scaling and resilience your own sites and applications. Each one of these sessions typically lasted an hour, and as the event was completely free to attend, gave attendees a great insight into how open source is being used.

We really enjoyed our brief time on Jersey, and we think the event itself was a great initiative that

has hopefully given some local businesses another option when it comes to software. That this event could be organised by a such a small group of people is a testament to their commitment to open source, and we sincerely hope they're able to do the same next year.

TELL US ABOUT YOUR LUG!

We want to know more about your LUG or hackspace, so please write to us at lugs@linuxvoice.com and we might send one of our roving reporters to your next LUG meeting.

London PostgreSQL meetup

Josette Garcia reports from the world of alternative databases.

On 21 January, 35 members of the London PostgreSQL Meetup met in the offices of Mind Candy near Old Street (near the famous Silicon Roundabout). In case you are unaware, Mind Candy is the creator of the *Moshi Monsters* games for children. You might have taken your kids to *Moshi Monsters: The Movie*. The company has now created the *World of Warriors* game, which seems to be for a more grown up audience.

Not only did Mind Candy offer the use of its premises, but it also provided pizzas and beer which made it a great opportunity to meeting some new and old friends from the Python community.

After pizza and beer, the real session started with Howard Rolph on the key features of *PostgreSQL 9.4*, which are:

- JSONB Binary JSON storage.
- Replication improvements.
- Changeset streaming/logical decoding.
- Performance improvements.
- Alter System.
- Refresh materialised view concurrently.
- Backwards compatibility.

After this the talks continued with



Rach Belaid, with a session entitled 'Postgres Full-Text Search is Good Enough!'.

Rach, who is the co-founder of web and iOS agency Lost Property, describes himself as passionate about open source and web standard technologies, and believes in an open and collaborative working environment to increase performance, the quality and the sharing of knowledge. He specialises in Python, Pyramid, Unix/Linux based systems and uses *PostgreSQL* when in need of a



database management system. I found it highly encouraging to see the audience participation – questions were asked, little debates took place – all in all it felt that the talks were worthwhile and everybody learnt a little or more.

I believe the next meetup will happen in April and will focus on Perl, but don't take my word for it – keep checking www.meetup.com/London-PostgreSQL-Meetup-Group if you're in the area and curious about how the heavy lifting of the web happens.

This looks to be one of the best locations we've seen for a geek gathering. Tropical forestation and bean bags anyone?

Ipswich Makerspace

Tim Elliot reports on the hacking going on England's eastern shore.

Our makerspace is a place where like-minded people can get together and share their interest in science, electronics and all things technological. The aim is to share knowledge, show off our skills and inspire each other to new heights of inventiveness.

We meet twice a month, once for personal projects and once for talks, show and tell and group discussions. A chance to learn something new like PCB making or hear about the latest tech from someone who's been hands-on

with it. We have microcontroller collectors, quadcopter pilots, robot builders, several members who own 3D printers, and vast numbers of unfinished projects.

We meet in a church hall that we rent for the evening, and hope soon to have a home of our own, where we can keep fancy-pants equipment for joint use.

Flushed with the success of our Tractorbot team at Pi Wars in December, where we won the Under-£75 category (1st in Obstacle Course, 2nd in Three-Point

Turn, came within a gnat's crotchet of winning Proximity Alert), we are launching a series of mini-courses, to include PCB making, PIC programming, SQL, Timelapse photography, soldering and more.

Ipswich Makerspace (Facebook <https://www.facebook.com/groups/ipswichhackspace>, Blog <http://ipswichmakerspace.com>) meets every 2nd and 4th Thursday, 7pm, Trinity Church Hall, Back Hamlet, Ipswich IP3 9AJ, Suffolk. Your first meeting is free! 📍



Our good friends at the Free Software Foundation Europe were there, reminding us all that the freedom in FOSS is more important than merely the practical benefits.



Larry Wall gave a hugely entertaining presentation on Perl 6 – we'll have an interview with the man in a future issue!



FOSDEM 2015

This yearly geek-fest in Brussels brings together free software developers for coding sessions and beer. **Mike Saunders and Ben Everard were there.**

FOSDEM has been running annually since 2001, and is arguably the best European meetup for developers of free and open source software. It's free to attend, there's a packed schedule with presentations and demos, and it's fortunately not teeming with buzzword-bleating suited business types trying to sell you data silo-enabled Web 3.0 cloud container internet of things devices, or other nonsense like that. No, this is the place you go to meet real geeks: the people working on Debian, Fedora, *Firefox*, *LibreOffice*, the kernel, Systemd and other pieces of software with which we're all familiar.

And it's big, spread across several buildings of the Université libre de Bruxelles. Part of the complex is reserved for stands, where FOSS projects can demonstrate their latest wares and talk to interested passers-by. The Debian stand, for instance, showed a braille typewriter and screen reader being used by a blind developer, highlighting how the distribution really strives to be a "universal" operating system.

Along with big names like Fedora and *Firefox*, some smaller projects were also represented. We were happy to meet up with the ReactOS team, which is working on an open source Windows-compatible OS

(www.reactos.org). It's an obscure project and doesn't get much attention, but it runs an impressive range of (predominantly older) Windows software, and could help some people make the transition from proprietary to free software.

All change

One stand that really grabbed our attention was that of OpenMandriva LX, a desktop distribution spun-off from the once-famous Mandriva. The developers had an ARM board connected to a keyboard, mouse and monitor, with a sign above saying that the distro had been "100% compiled with *LLVM/Clang*".

The primary reason for switching to this new compiler over trusty old *GCC* is compilation speed. One developer explained to us that building the distro's 15,000 packages takes half the time using *LLVM/Clang*, when compared to *GCC*, and the newer compiler also produces better code for low-spec ARM devices (such as the board being demonstrated). While we were there, we also asked the OpenMandriva team if they could ever consider merging with Mageia, another Mandriva fork. We were told that there are no bad feelings between the projects, and a merger was

"FOSDEM is arguably the best European meetup for developers of Free Software."

Many big-name distros were present, including OpenSUSE, Fedora and Debian.



even suggested a while back. Today, though, the distros have differing goals, so we probably won't see a merger any time soon. Also on the subject of compilers, we attended a talk by Behan Webster, who is leading a project to make the Linux kernel compilable by *LLVM/Clang*. He noted that the compiler is getting up to par with *GCC* in terms of code size and speed, and is improving rapidly. *GCC* has come under fire recently for being monolithic – ie it's hard to use parts of the compiler toolchain in other projects – but according to Richard Stallman, creator of *GCC*, this is important to make sure it doesn't just end up as part of a larger, proprietary IDE.

LLVM/Clang is more modular and has a more permissive licence, which is preferable to some users. It also has other benefits, such as a built-in static analyser for C, C++ and Objective C programs. Webster noted that building the Linux kernel with *LLVM/Clang* brings other benefits, such as making sure

that the kernel code remains fairly standardised and not dependent on the quirks of a particular compiler.

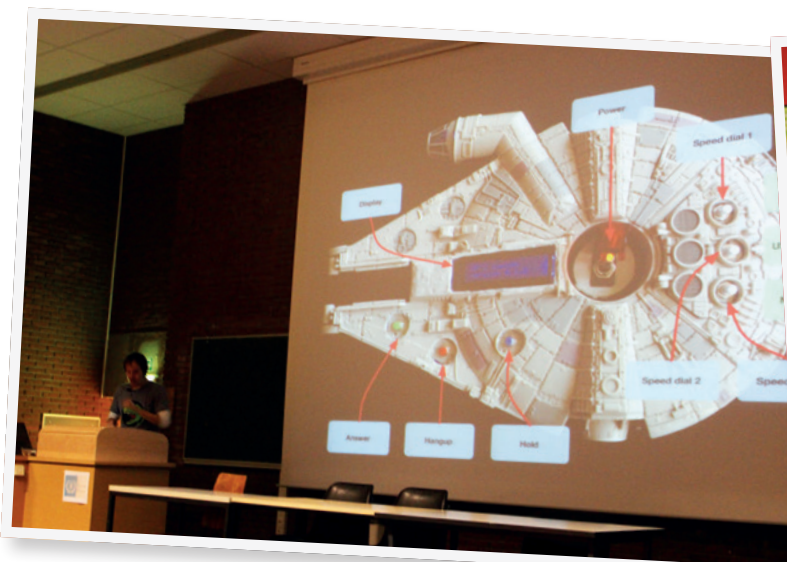
We attended a few of the lightning talks – 15-minute presentations that attempt to woo you with new ideas and technology. The developers of Crazyflye, a tiny quadcopter (www.bitcraze.se/crazyflye-2), entertained the audience by flying one around the arena, while the OP^2 Raspberry Pi-powered phone won geek points despite its un-Googleable name.

But our favourite talk of all was from Larry Wall, the lead developer of the Perl programming language. Wall is a fascinating chap, having a background in linguistics, and his presentations about Perl are always full of jokes and references to films and books. Perl 6 has been languishing in development hell for years, and Wall has often quipped that "Perl 6 will be ready in time for Christmas – we just don't know which year."

At FOSDEM, Wall delivered a bizarre and witty talk comparing the Perl 6 development process with *The Lord of the Rings*, and ended by saying that, yes, Perl 6 will be released in time for Christmas. But this time he actually stated a year: Christmas 2015. This brought a standing ovation from the crowd, and we managed to grab Wall for a chat afterwards, so stay tuned for an interview in a future issue. 📺

FOSDEM is always a great place to pick up merchandise, such as T-shirts, mugs and stickers.

Below left Saúl Ibarra Corretgé demonstrated a funky-looking open source hardware VoIP phone, powered by a Raspberry Pi. **Below right** O'Reilly had a table of its latest books, with some mightily hefty tomes on coding, networking and sysadmin.



Inside the...

UBUNTU PHONE

The Ubuntu phone is here at last! Explore its development, its features and what it means for Canonical and the Ubuntu desktop.



Almost as soon as the first version launched in 2004, Ubuntu permanently changed the Linux distribution landscape. 2004 was a time when the desktop was still important, and Ubuntu presented the Linux desktop not as alien territory, only to be ventured through with the right skills, but as a verdant pasture of adventure and possibility. As its 2004 tagline proudly proclaimed, this was Linux for Human Beings, and it enabled millions of people to use Linux who may not otherwise have done so.

Under the aegis of its parent company Canonical, Ubuntu is still a huge success. It's now the distribution that non-Linux users will most likely have heard about, or have even tried. It's used when migrating offices and local councils to Linux, and it's used in many servers and cloud instances. It's also the basis for many other popular distributions,

including Mint, gNewSense, Google's own derivatives and the semi-official KDE, Xfce and Gnome versions. Its easy installation and no-nonsense approach to adding applications or upgrades has forced every other distribution to up their game, and it's helped make the Linux desktop a viable alternative to OS X and Windows.

“Canonical needs to capitalise on its success and mind-share and make more of its influence.”

But Canonical is facing something of an existential crisis. It needs to capitalise on its success and mind-share and make more of its influence. This is happening in the cloud, with Ubuntu finding favour as the first choice behind many servers,

but Canonical also recognises that it needs to diversify.

Which is where the phone comes in. First touted as a cutting-edge convergence device, and the focus of a hugely ambitious crowdfunding campaign, the first incarnation of the Ubuntu Phone is here. And we've got one.

Inside Canonical

Why phones? Why now?

Canonical employs around 600 people. This makes it tiny in comparison to other phone manufacturers. Samsung alone moved 1,000 of its employees to work on Tizen, and it could do this almost overnight. Canonical doesn't have that kind of infrastructure of funding. And it's not a phone manufacturer.

But there's more to Ubuntu than a popular Linux distribution. It's also the most visible facet to a company strategy trying to generate income from open source. The distribution was famously founded by the multi-millionaire space tourist South African Mark Shuttleworth, after he pooled the initial team from mailing lists he read while free of the internet on an icebreaker. And while there is an Ubuntu Foundation to ensure the longevity of the distribution itself, the distribution is at the heart of a business he also founded, Canonical.

Like Red Hat, Ubuntu is also used widely as a server operating system, and more recently in the cloud, with a reported 64% of OpenStack deployments – and it's even popular on Microsoft's Azure cloud platform. But Canonical makes very little money from all these people spinning up instances of the world's favourite operating system. This is open source, after all, and there's nothing forcing anyone to pay for anything, even when those instances are dialling back to Ubuntu's servers for updates and upgrades.

Ubuntu is undoubtedly a huge and growing success. But it's also true that Canonical has yet to tap into the revenue potential of its own operating system, and it's struggling to make a profit. Last year's financial report on its performance 2012–2013 showed a loss of \$21,343,00, despite gross profit being up from \$54 million to \$61 million. And this is where the requirement for a new direction steps in, and why 2015 is going to be pivotal for its future and the future, health and investment in the Ubuntu operating system.

Newbuntu

Ubuntu is used everywhere, but money-making potential has remained elusive. To solve that problem, Canonical needs a piece of its own turf, one that it can invest in, capitalise on and hopefully make money from. And that's exactly what it has spent the last two years creating – not as a single

technology, but as a swathe of innovations plugged into the heart of its operating system, from the desktop to the cloud. It's had to sacrifice its standing within the community to do this – moving away from Gnome and Wayland, for instance. But this has been part of its strategy for staying in control. Canonical is transforming the way Ubuntu is put together and used. And the first real, physical and tangible step towards making this a reality is the launch of the Ubuntu Phone.

The Ubuntu Phone is the most exciting development to come out of Canonical for

years, and its relevance for both Ubuntu and Canonical is something called convergence. Originally, convergence meant plugging in your phone and continuing to work with a keyboard and screen. It now means use the same interface on multiple devices – hence the redevelopment of Unity and the Mir display server running in the background – and that's a difficult trick to pull off. Microsoft failed spectacularly by trying to augment Windows 8 with touch-friendly characteristics, despite almost no one being interested in using a touchscreen with their Windows laptops or PCs.

Ubuntu phone hardware: Two different makes and models



BQ Aquaris E4.5

CPU 4-core MT6582 1.3GHz
GPU Mali 400 500MHz
RAM 1GB
NETWORK 802.11 b/g/n, GSM/HSPA
STORAGE 8GB
SCREEN SIZE 4.5 inches
RESOLUTION 540x960 – 240ppi
CAMERA 8MP (rear) 5MP (front)
DIMENSIONS 137 x 67 x 9 mm
WEIGHT 123g
BATTERY LiPo 2150 mAh
CONNECTORS Dual micro-SIM, micro-USB, headphone jack, MicroSD (up to 32GB)
PRICE €169 (only available in Europe)



Meizu MX4

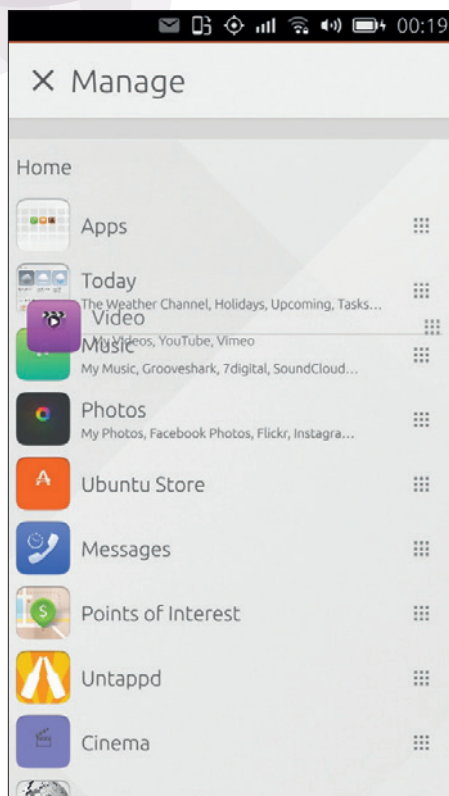
CPU 8-core MT6595 1.7/2.2GHz
GPU PowerVR G6200 MP4
RAM 2GB
NETWORK 802.11 a/b/g/n/ac, GSM/HSPA/LTE
STORAGE 16/32/64GB
SCREEN SIZE 5.36 inches
RESOLUTION 1152x1920 – 418 ppi
CAMERA 20.7 MP (front) 2MP (rear)
DIMENSIONS 144 x 75.2 x 8.9 mm
WEIGHT 147g
BATTERY 3100 mAh
CONNECTORS Micro-SIM, micro-USB, headphone jack
PRICE TBA

Inside the phone

2014 – the year of Ubuntu on your phone?

Launching a mobile phone with a new operating system in 2015 is crazy. People of Earth carry over a billion Android devices alone, and Android is fundamentally an open source operating system, negating the moral imperative for creating another. Forking Android has been shown to work too, at both ends of the scale, from Amazon to Cyanogenmod. And we won't mention other open source alternatives like Jolla, Tizen or Firefox OS. But that doesn't mean someone else shouldn't try, and there's something intrinsically brilliant about open source in that it lets projects succeed or fail judged by their own merits. For Canonical, that means a strong emphasis on open source, open platforms and fundamentally, choice.

The turning point for Canonical must surely have been the Ubuntu Edge crowdfunding campaign in 2013. It was ridiculously optimistic: Mark Shuttleworth was asking for \$32 million to give Canonical the cash to build a cutting-edge smartphone running its own operating system. Of course, the campaign failed.



Scopes are how you interact with the Ubuntu Phone. They can be installed, removed and their order shuffled around, but they're always there.



Both Canonical's CEO, Jane Silber, and its VP of Mobile, Cristian Parrino, gave an impassioned talk about the importance of the Ubuntu Phone at its launch in February 2014.

But it was a spectacularly winning failure: \$10,267,352 was pledged from more than 22,053 contributors, making it the largest crowdfunding campaign of the time. And whether this was a publicity stunt or a genuine attempt to fund a new phone platform, there's no doubt it left Canonical with the very real desire to create a phone.

Two years later

Two years later, we find ourselves in London on a chilly morning in February. We're sitting with approximately 40 other people in a hotel in London. This is an 'Insiders Event' where the long awaited Ubuntu Phone is going to be revealed in partnership with BQ, a major phone and tablet manufacturer from Spain.

That Canonical isn't launching the phone after a campaign of tantalising leaks and a conclusive fireworks display at Mobile World Congress is a significant sign that Canonical knows it can't compete with the likes of Samsung. Its Ubuntu Phone is going to need to attract a different kind of customer. This is likely the same reason why early batches of the phone we're about to get our hands on are sold 'flash sale' style to try and generate as much interest as possible. BQ has said it was handling 12,000 orders per minute during those initial flash sales, and selling out within 10 minutes, but there are no specific numbers available on the final quantities that have been sold.

Most people at this event are vocal community members, or people who have helped Ubuntu in some way. They're not always the people with the largest number of followers on Twitter, or YouTube. They're people with a genuine enthusiasm for Ubuntu and Canonical, and this sincerity is what comes through from the beginning, when the announcement finally comes.

"There are no words to describe how excited I am, and the rest of our colleagues, the engineers, our CEO, people who do the design, Mark..." says Cristian Parrino, VP of mobile at Canonical, by way of a very emotive introduction. He goes on to talk about not bringing another app-centric mobile phone platform to the market, about giving users a richer, less fragmented experience, "but most of all, more personal."

The origins

For us, Ubuntu's netbook remix is the starting point of what has become Ubuntu Touch and Ubuntu Phone. It was here that the first pixels of what would become Unity made an appearance – a launch bar down the left of the screen, and a frugal use of display real-estate. This was followed by a migration to full-screen applications and unified menus and finally, the idea behind scopes. The scopes idea is Canonical's great hope for the Ubuntu Phone, because it's what it hopes will differentiate its operating system from the competition, and it's what

we first played with when we finally got the phone in our hands.

Canonical hopes that scopes will differentiate their efforts from those of the competition, and perhaps, justify its commitment to both the Mir display server, already running on the Ubuntu Phone, and the Unity desktop interface. Scopes are tied closely to Unity and Canonical's convergence strategy – using the same user interface and even sessions across multiple devices, which is why Canonical has gone it alone with so many of the APIs behind the desktop.

Scope for improvement

However, the first and biggest problem with scopes isn't a technical one. It's explaining what they do and why they're potentially so powerful. This isn't so much a problem on the desktop, where we've got used to scopes as a way of switching between different kinds of search result. However, it's not always clear what advantage this offers over a sorted list of results – where images or music files appear separated from other documents that satisfy the search criteria, for example.

The answer is that the results are aggregated from various sources. For music, that might be your local music collection, an online service and perhaps a store. For news, that might be the top stories

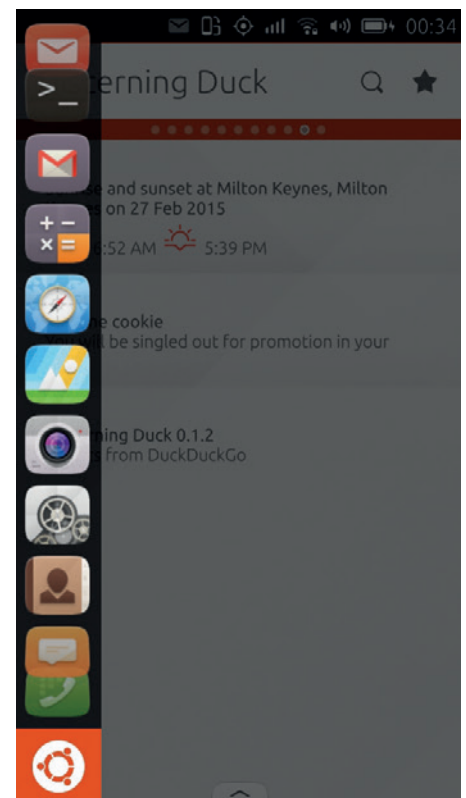
from a selection of websites. But getting this message across to users of a new smartphone is going to be a challenge.

Scopes on the Ubuntu Phone are the default view. They're what Canonical wants you to use to get the most out of your device and they're launched with the easiest screen gesture to pull off – swiping from the left edge of the display into the middle of the screen. This is initially confusing, because this same gesture also displays the launch panel, a vertical list of running and quick-launched apps that's functionally identical to the desktop edition. Continue swiping and the currently running app is slid to the side, revealing whichever scope you were running previously. The first scope is labelled 'Today', and it's the perfect example of the kind of data scopes pull into a single window.

The Today scope is Ubuntu's equivalent to Google Now, only the information it pulls together to show on a single panel is totally under the user's control, and far more comprehensive. And unlike Google Now, the developers have complete control over what information is aggregated and how, rather than relying on Google's dark magic and an open invitation to raid your web browsing history. The Today scope shows the date, the local weather and upcoming events, as well as phone-specific events such as recent calls and messages, for instance. It also pulls in data from

“You cannot bring a phone to market by turning out another app-centric interface.”

external services, listing Twitter trends, for example, or the latest new stories. Scopes are enabled and disabled by using stars in the top-right of each view.



Just like the panel on the Ubuntu Desktop, you can pin applications and switch between those that are running on the Ubuntu Phone.

Pressing the 'Configure' icon in the top-right alongside the star will enable you to choose elements you want enabled or disabled. For Today, that means a list of 15 different sources, from upcoming holidays to *FitBit* stats. It's this kind of aggregation that's key to how scopes work and why they could potentially be more effective than running a single app for a single task.

They're not that dissimilar in function to the user interface of the hugely successful Pebble Time, which has just been successfully crowdfunded. With the Pebble Time, rather than making its users launch specific apps for specific functions, it takes nuggets of information from various app and data sources and presents these on a timeline that stretches from the past and into the future – just as you might expect with a watch.

Scopes can do the same thing, only they are most useful when there's some context, such as pubs close to your location, major news stories or Wikipedia entries for sites close to where you're staying, and they're what makes the phone so interesting to use.



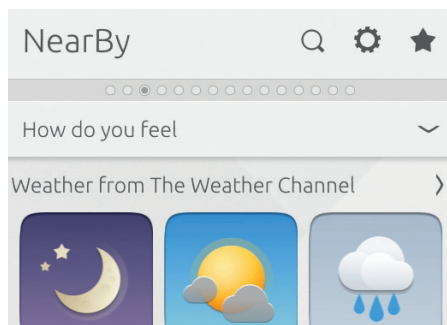
The phone's web browser is derived from the same rendering engine used in Google's *Chrome*.

Some of our favourite scopes

Forget apps for now – these are what Canonical wants you to be impressed with.

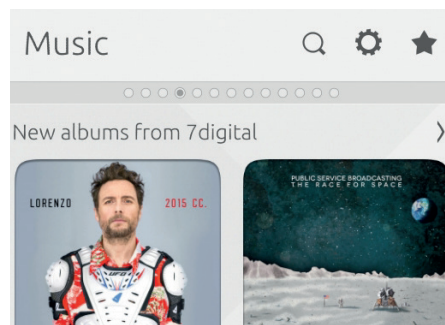
NearBy

If there's one example that best epitomises the idea behind scopes, it's this. Taking your location as a starting point, this scope populates itself with music, photos and restaurants that have some link to where you're currently standing. A drop-down menu also lets you choose a mood. Telling NearBy that you're thirsty will return a list of bars (a fish bar was top of our list); if you say you're stressed, you'll get the location of your local spa, some relaxing music suggestions and a list of games. It will even pull in information from other related scopes, such as local Wikipedia entries.



Music

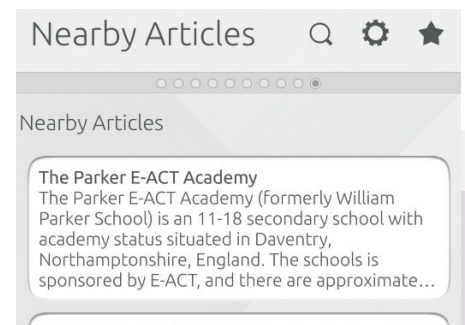
The Music scope lists results from several mainstream music providers, including 7digital, SoundCloud and YouTube. Many of us listen to music from more than one source, and a scope for managing your access to those sources when you just want to listen to something makes better sense than opening separate YouTube or SoundCloud apps, but the back-end is a little too limited at the moment. The other problem is that the security lockdown on the device doesn't let third-party apps play music in the background. Photo and video scopes offer similar facilities.



NearBy Articles

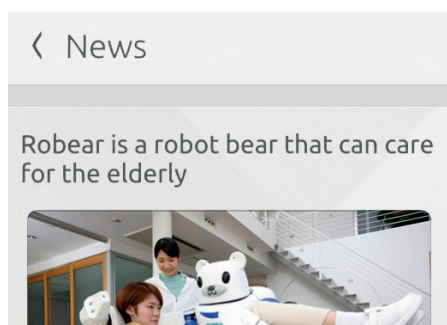
This is such a simple scope – it provides a single paragraph for Wikipedia entries that have a geographical location close to your current position. But it's brilliant. You often find yourself updating the scope even when you're driving through somewhere that looks interesting. If you need to know more, click on the link to open the web browser.

If you wanted to replace NearBy with this Articles scope, you can swipe up the scope configuration panel, hold down on the scope you want to move, and the management view will appear, allowing you to drag and move the position of any of the scopes.



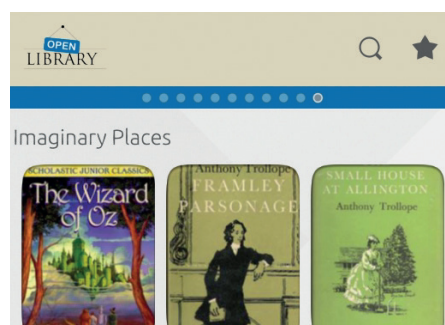
News

The News scope is another powerful example of scopes working well, giving you a lot of control over what kinds of stories (and their sources) are delivered to your device. An RSS feed is presumably the source for this data, as there's only a paragraph and a single image to accompany the stories, but it's enough to give you an overview of what's happening and how those stories are being reported by the different media outlets displayed within the page. The only serious omission is the ability to add your own sources, but there are other RSS readers for that purpose.



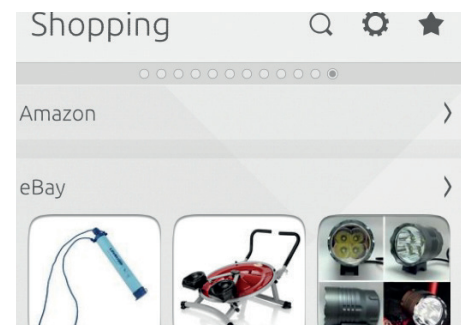
Open Library

Many of the applications and scopes that can be downloaded from the Ubuntu Store are open source and their licence is an important part of the information you're presented with before download. Open Library is one of the many open source applications that feels like an online store but it's actually listing books that you can legally read for free. Most of these are classics, but the Open Library also lets you borrow digital books, as well as download those that are out of copyright, often as PDF, HTML and ePub.



Shopping

This scope aggregates products in the same way that other scopes aggregate music or news. Default sources include eBay, Amazon and Etsy, and it could potentially be a great way of listing the same products from different sources so that you can compare prices and services. This is what it does when you use the search field. But it could be expanded to do so much more. The GPS could be used to list useful products when your phone knows you're away – such as umbrellas in London – but it could also list competing prices or products when it knows you're in a specific store or looking for a gift.



Gesture control

How the Ubuntu phone is innovating in user interaction.

At the top of each scope panel, there's a small breadcrumb trail of dots, which are used to represent which scope you're currently looking at. Swipe left or right across this small section, or any blank section of background, and you swipe between scopes in the same way you might swipe between virtual desktops.

One of these scopes is called 'apps', and this is where users of other phones will feel at home. This scope behaves exactly like the app launcher for Android and iOS, and includes access to the Ubuntu Store and some integral functions like messaging, the camera and phone. The app icons can't be manually rearranged, but they can be limited by category and pinned as a shortcut to your launch bar, just as you might on the desktop.

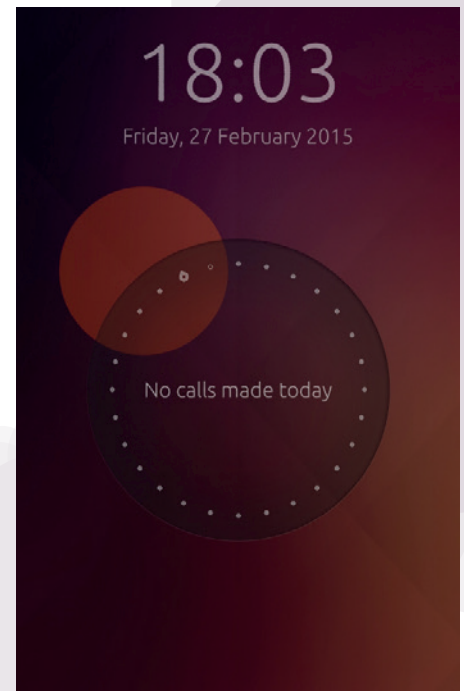
Left, right, up, down

One of Canonical's other innovations is the use of every screen edge to trigger a gesture. As we've seen, dragging in from the left edge will first show the launch panel before swiping away the currently running application to reveal the scopes interface. Swiping from the right edge is the equivalent of launching the task switcher. Every application you've got running is concertinaed across the screen, enabling



You can always swipe from the right screen edge to quickly switch to the previous task or open the task manager.

you to select them or flick to close them. A quick swipe is a shortcut to the previous application you were running. One feature unique to the Ubuntu Phone is that if you



Those dots and circles on the unlock screen are used to tell you how many things your phone has done today – such as photos, or messages.

continue to hold one of these gestures and reverse your motion, the gesture is cancelled. For task switching, that means you can see what's running and slide back to your original app without any interruption. The same principle is used for the notifications and quick settings panel, which is pulled down from the top border.

The panel you see will depend on where your finger is located horizontally across the top of the screen when you initiate the drag. On the far left, you'll get the notifications list, while on the far right you'll be able to configure the date and time. Between these points, there are panels for rotation, files, location services, Bluetooth, networking, sound and battery life. But if you hold your finger down and move to the left and right, you can switch between these modes dynamically and even close the panel without performing a single function.

The final edge – sliding up from the bottom of the phone – is used to open a contextual menu. The contents of this menu change depending on what application you're running. From any scope, for example, you can use this menu to enable, disable and install other scopes, while the Call function uses the menu to list recent calls.

Get developing!

Getting developers to write new applications for a new platform is fundamental to its success. As Cristian Parrino put it when introducing the phone, how to attract developers "is the quintessential question." And considering Google has only just started to get serious with its own development environment – *Android Studio 1.0* was only released in December 2014 – Canonical has already made great progress by providing a fully fledged development environment. The Ubuntu SDK is easily installed from any Ubuntu desktop and it includes the development libraries, an emulator for testing code without any hardware, and a graphical development environment. You can also perform all kinds of remote tasks on your real Ubuntu Phone, such as connecting securely via an SSH session.

The application at the heart of these development tools is the venerable *Qt Creator*, with a few modifications to act as a portal for Ubuntu Touch development. The reason for this choice is QML, the scripting framework that takes the best bits of JavaScript (ubiquity and speed) and binds them to the expansive *Qt* user-interface library. This should enable almost anyone to build fully

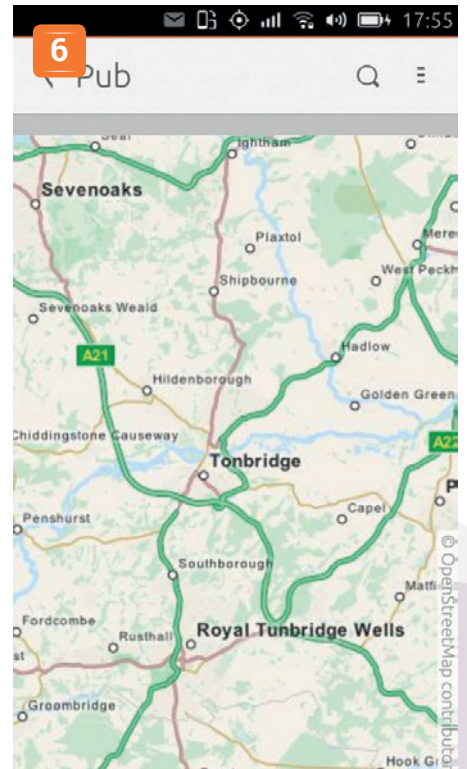
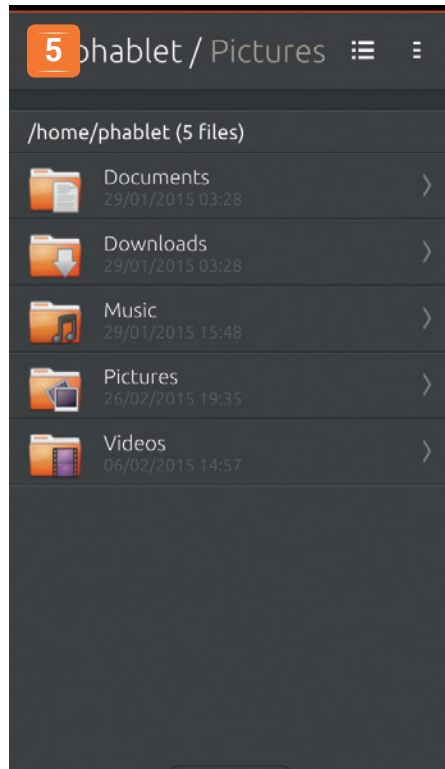
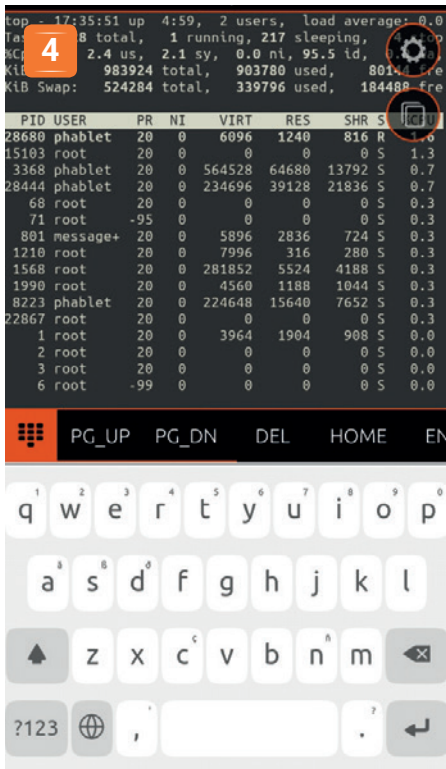
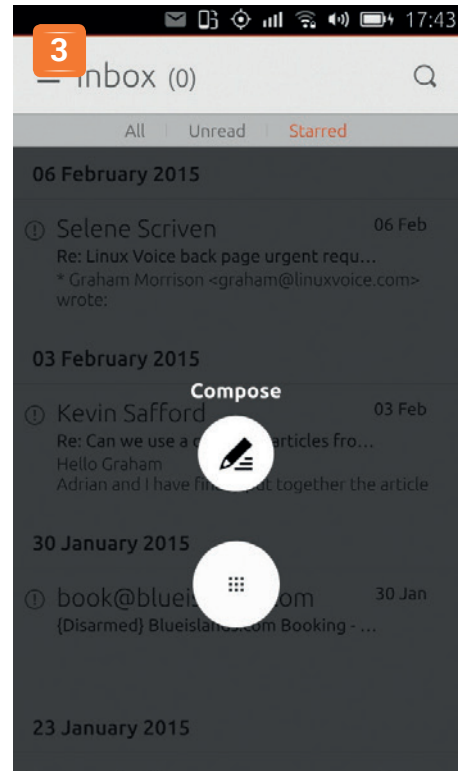


The development environment for Ubuntu Touch applications is a customised version of the exceptional *Qt Creator*.

functional applications without too much difficulty. Another option is to use HTML 5 to develop your applications. Many of those that are bundled with the phone, and the desktop, do exactly this, and it means you can create truly cross-platform solutions from the same codebase.

Applications

Our favourite apps for the Ubuntu Phone OS.



1 Camera We take more photos than we make phone calls with our phones, so this app is important. **2 Cut the Rope** There are many games but not so many tier-1 titles. This will change as more developers get on board.

3 Dekko An ace email client built to use all of the Ubuntu Touch UI elements. **4 Terminal** This wouldn't be Linux if we couldn't access the terminal, and Canonical's own application is one of the best. **5 File manager** The operating

system isn't hidden from the users, but everything is strictly sandboxed. You can still modify your own files freely. **6 OSMTouch** There are a few options for navigation, but this is the best way of accessing OpenStreetMap.

Convergence

The future of Ubuntu – and some would say the future of computing.

“Convergence is the future of computing. So we've reshaped Ubuntu and combined the mobility of a smartphone and the power of a desktop on a single device.”

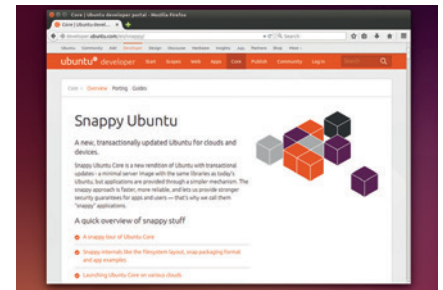
These were the words chosen by Mark Shuttleworth to start the promotional video that accompanied the launch of the Ubuntu Edge crowdfunding campaign back in 2013. And despite convergence being a difficult word to market, lots of potential users got excited by the idea of connecting a keyboard and screen to their phones to work more productively, just as you would on a desktop PC or laptop. The Ubuntu Phone doesn't have these features, but convergence was still an important part of the launch presentation. But the emphasis was different. Mark, for example, mentioned convergence as the unification of x86 and ARM – the combination of laptops with mobile phones. But this isn't likely to be from the same devices, and is more likely to be a feature that enables you to continue using the same application or workspace on more than one device.

Ubuntu has been reshaped too, as Mark originally promised. Scopes are an integral part of desktop Unity, even if they're not as developed or as diverse as those that appear on the phone. And while the single-device

Snappy Core and the cloud

One of the best things to come out of Ubuntu Touch is Snappy Ubuntu Core (see issue 12 for our FAQ). Snappy Core is a minimal version of Ubuntu along with a cloud-focused package manager that makes it easy for sysadmins to create new services and spin them out across lots of instances or servers. Like Docker, each application is isolated, self-contained, sandboxed and secure – a development that only came about because Canonical needed a self-contained, sandboxed and secure solution for installing applications on Ubuntu Touch.

According to the OpenStack Foundation global survey, Ubuntu is the most popular host and guest operating system, with more than half of all OpenStack instances running Ubuntu, an even larger proportion for public clouds. If Snappy Ubuntu Core can help Canonical turn some of that popularity into profit, Ubuntu will be in an even



Snappy Core and Ubuntu's success in the cloud are both positive side-effects of investing in new ideas.

stronger position, as will the many users who cut their teeth with Ubuntu as a first distribution and want to find work within the industry.

features touted for the Ubuntu Edge aren't in Ubuntu Touch today, they may not be far away. Ubuntu Desktop Engineering Manager, Will Cooke, has prepared a demo running on both Intel and ARM tablets running Ubuntu Touch, where applications pop-out of full screen and into a windowed mode when you connect a wireless mouse, and you can run desktop applications like *LibreOffice* and finally connect to a real screen, just as Mark

Shuttleworth said you'd be able to do. The Unity, Mir and Xmir code needed to perform these tricks isn't quite ready yet, but it looks like it's not going to be far off, which won't affect the modest BQ Ubuntu Phone, but it will open new possibilities for convergence on faster tablets and phones, as well as the Ubuntu Desktop itself.

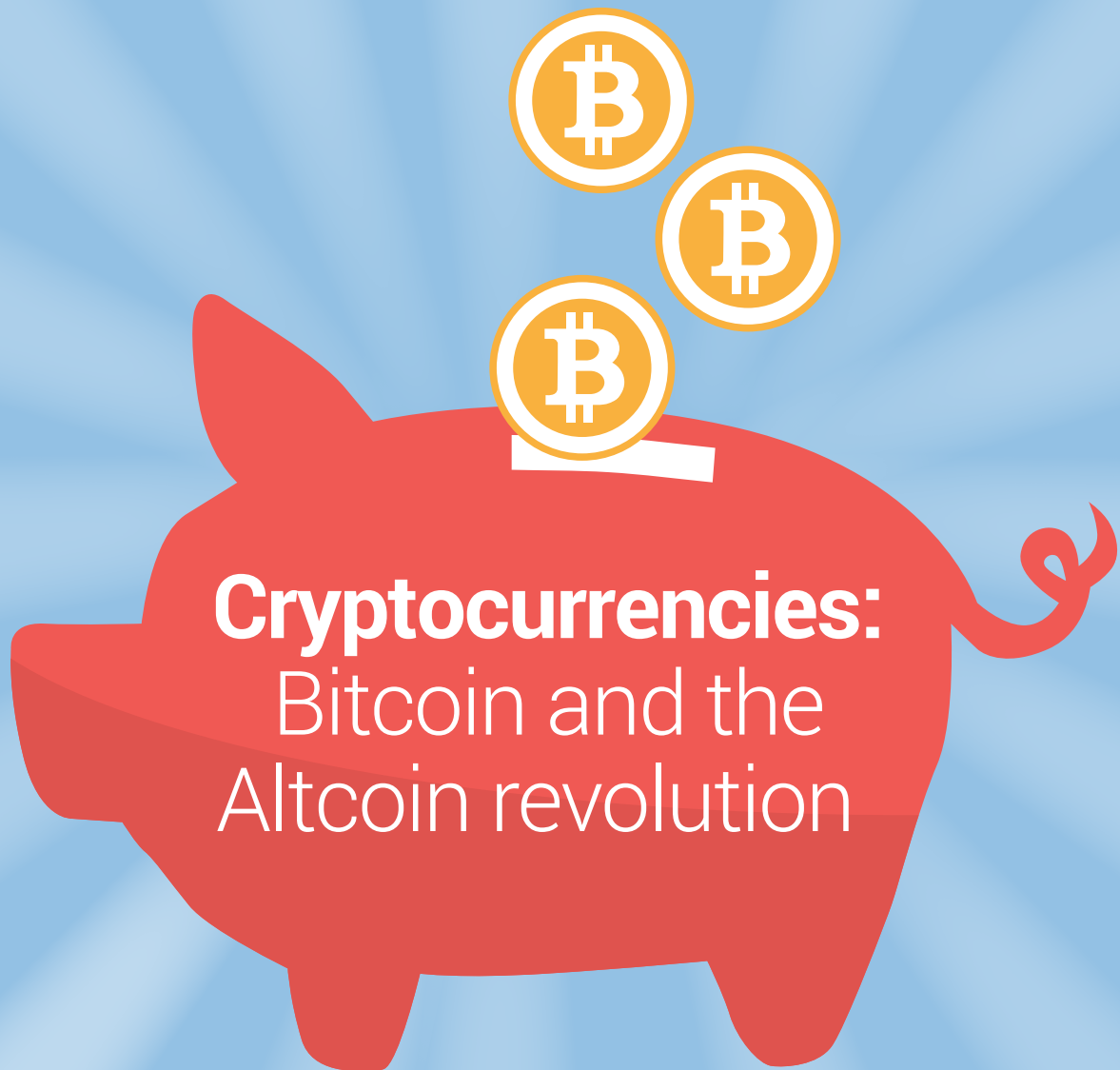
A future full of choice

Ubuntu Phone, Ubuntu Touch and the Unity desktop are all part of Canonical's strategy to put Ubuntu into a stronger position. If this succeeds, it will mean the future of Ubuntu is assured, even if the desktop becomes less relevant through more convergence with other devices.

But most importantly, it offers choice. As Jane Silber said when speaking at the launch of the phone, “We're not at the end of what personal computing looks like.” In many ways, we think we're still at the beginning. iPhone and Android are winning the current round, but we all know how quickly things can change, and we're happier in a future where companies like Canonical try new things, than a future where they accept the status quo and things stay the same. This is what's so good about the emergence and the final release of the Ubuntu Phone. It takes what started as an easy alternative desktop operating system and pushes it into our pockets – and that's something to get excited about. 



Core applications include the web browser, the gallery, a note taking tool and the media player.



Cryptocurrencies: Bitcoin and the Altcoin revolution

There are now hundreds of cryptographically secure currencies, but why do they exist and which ones should you trust?

Currencies backed by cryptographic guarantees rather than by governments or precious metal stores first became famous with the dramatic rise of Bitcoin in 2013 when one Bitcoin rose from a value of \$14 to over \$1,000. The last year has been a bit less impressive and the price of Bitcoin slumped to about 20% of its peak.

Despite the low price, there's good reason to be positive about Bitcoin. The last year has seen the currency become better known and more useful than ever. You can spend it in more places, and Linux Voice subscribers can renew their subs with it (we hope to roll out sales for new subscribers soon). Bitcoin is by far and away the most popular cryptocurrency, but there are hundreds of

others, known as altcoins. Some of these altcoins are gaining popularity, while others are languishing without value and without miners to keep the blockchain moving. Some of these new

cryptocurrencies hope to add new features, or improve on the Bitcoin model in some ways; others are just scams perpetrated by people hoping to get rich quick.

“The last year has seen Bitcoin become better known and more useful than ever.”

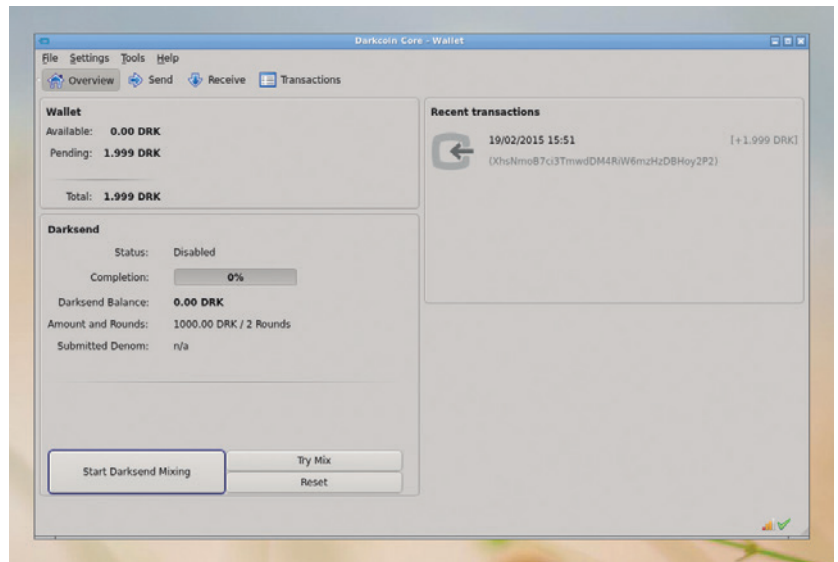
Almost all cryptocurrencies work in the same way – the method pioneered by Bitcoin. Miners calculate new blocks that are added to a cryptographically signed list that goes back to the very beginning (this list is known as the blockchain). Transactions are added to the blockchain, and once there, they're an irremovable part of the currency's history. This

permanent ledger of every transaction prevents both double spending and making fake coins. Anyone can inspect the blockchain and make sure that the coins are valid (that is, they can be traced back to the point they were mined) before making a transaction.

However, despite working in the basic same way, there are some important differences between the currencies. Perhaps the biggest distinction from a technical point of view is the hashing algorithm used to mine and secure the blockchain. Some of the most popular are:

- **SHA256** Used by Bitcoin. This algorithm is now implemented in highly efficient ASICs (see boxout on mining), so it's no longer possible to mine it efficiently without purchasing specific mining hardware. There is a slight risk that this could lead to a small number of people getting control of a large amount of the hashing power (by limiting access to hardware). However, currently this isn't happening
- **Scrypt** Originally this was thought to be resistant to ASIC (chips built for the sole purpose of creating coins) miners, because it requires more memory than SHA256. However, there are now Scrypt ASICs that can mine more effectively than GPUs. The difference isn't as great as with SHA256 though. This is the hashing algorithm used by Litecoin.
- **X11** This isn't a single hashing algorithm, but a collection of 11 different hashes chained one after the other. The theory is that this complexity will make it harder to design specific hardware to perform the hash effectively, and that this will slow down the development of ASICs and keep the mining more democratic for longer. At present, there are no ASICs that can mine X11 (though some vendors erroneously claim that they do). However, it is likely that if an X11 coin becomes valuable, ASICs will follow. The most popular X11 coin is Darkcoin.

Coin miners are constantly mining new blocks. The number of blocks mined since a transaction was included in the block chain is the depth of the transaction (sometimes called the number of confirmations). The deeper the transaction, the harder it is for anyone to reverse it. It's common to say a transaction is verified once it reaches a depth of six blocks in Bitcoin. Each coin has an algorithm that adjusts the mining difficulty depending on the



current hashrate in an attempt to keep the blocks being mined at a consistent rate, and the target rate is different for each coin. Bitcoin, for example, adjusts the difficulty to try and keep a new block appearing on average every 10 minutes. Since a transaction isn't valid at all until it's in a block, and not considered secure until it's in six blocks, it can take up to an hour for a transaction to be considered valid. This level of time is fine for some transactions, but it's not very good for, say, paying in a shop.

Hash rates and block times

Many other cryptocurrencies have faster block times. For example, Litecoin tries to get a new block every 2.5 minutes. This has two implications. First, it means that transactions are included in the blockchain faster, but consequently, it means that it's cheaper for a malicious user to manipulate a single block in the blockchain. The reason that blocks are considered secure in Bitcoin once they reach a depth of six is because at a depth of anything less than that an attacker with access to very powerful computers could try to out-mine the mining network.

The rules of Bitcoin say that the longest block chain is always the right one. Therefore if a transaction is included in one block, an attacker could start mining

Almost all cryptocurrency wallets are forks of the Bitcoin wallet, which means they have a *Qt* version that runs well on Linux. This picture shows the Darkcoin wallet.

Mining: can it become profitable again?

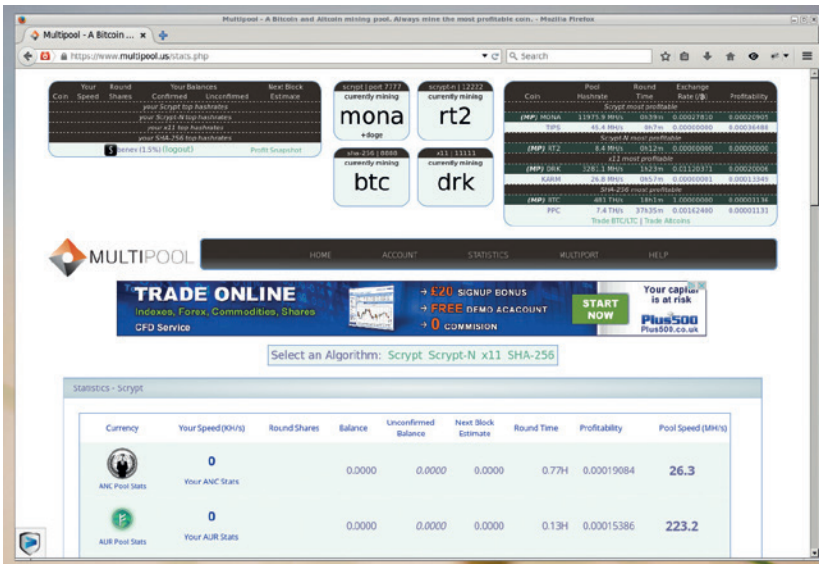
The original aim of mining was to distribute the task of generating the block chain to anyone with a computer who believed in Bitcoin, and so many of the early coins were mined on regular computers. However, as soon as Bitcoin started to become successful, people looked for ways to mine them more quickly.

Graphics cards can be programmed to mine the SHA256 hash quite effectively, and once software came out to allow this, it was no longer profitable to mine on CPUs (the cost of the electricity was more than the Bitcoin reward).

It didn't stop there though. People started to make hardware specifically to mine coins quickly. First, this was

using Field Programmable Gate Arrays (FPGAs) – these are blank chips onto which you can load circuits – and later using Application Specific Integrated Circuits (ASICs), which are custom-built chips. These days, it's not profitable to mine unless you have some of the latest generation ASICs and access to cheap electricity.

Currently, the best X11 currencies such as Darkcoin are right on the edge of being profitable to mine using a GPU. If there's an increase in price, this could mean that you can actually make money using your graphics card again, though FPGAs and ASICs will probably follow if mining X11 remains profitable for long.

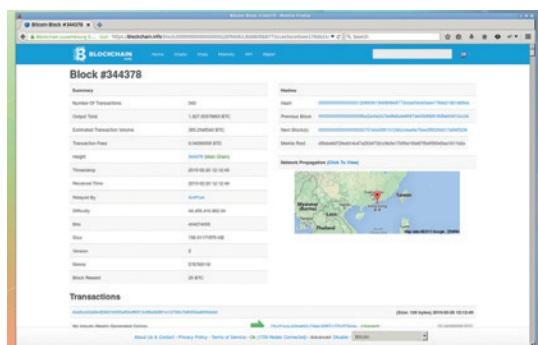


Some mining pools (such as multipool.us shown here) move between different cryptocurrencies depending on how profitable each currency is at the time.

on an earlier block, and if they can mine two blocks before the rest of the network can mine one, they can remove the transaction from the block chain even though it appeared in one block. The deeper in the block chain a transaction is, the more processing power they would need – and therefore the more expensive it would be. The faster block time on Litecoin means that an attacker would need fast hashing power for less time to reverse one block, so to get the equivalent level of security you need a transaction to be deeper.

However, many transactions are quite small. It's never going to be worth doing this to reverse a transaction for a can of coke or a pint of beer. For these smaller transactions a single block is enough, and that's going to be much quicker on average in the currencies with the shorter block times.

All cryptocurrencies give coins as a reward to miners. However, they manage this in different ways. Some have a large number of pre-mined coins that are for the currency's developers. Some have a fixed limit on the number of coins that will ever be created, while others will keep mining them infinitely. A large number of pre-mined coins (ie coins that were created before the currency went public) can be an indicator that the currency's creators want to enrich themselves rather than create a sustainable currency.



Using sites like blockchain.info you can see everything that's ever happened on the Bitcoin network.

What's not yet clear is the best approach to rewarding miners over a long period of time. Bitcoin halves the number of coins miners receive when they mine a block every four years. This means that fewer and fewer new coins will enter circulation as time goes on, and there is a limit on the number of Bitcoins that will ever be created – 21 million. The idea is that this limitation of supply will cause the value of Bitcoins to remain high.

On the other hand, Dogecoins will be mined forever. There is a risk here that these new coins will cause the currency to constantly fall in value. However, if growth in the Dogecoin market out-paces the new coins, it will mean that the coins will still raise in value and the miners will still be incentivised to mine. In currencies where there's a limit on the number of coins mined, there are often transaction fees (usually voluntary) that can be used to compensate miners when there are no more rewards for mining blocks.

In reality, for a cryptocurrency to be healthy, miners have to be paid. The falling block rewards and transaction fees model (like Bitcoin, Litecoin, Darkcoin and many others) mean that people who make transactions will pay the miners. In a currency that continually creates new coins (like Dogecoin), it's the people who hold coins that pay (because of the devaluation caused by the increase of supply).

Darkcoin

This covers most differences between most cryptocurrencies. However, there is one that's a little different: Darkcoin. This currency set out to fix what some people see a fundamental fault in the Bitcoin network: the lack of privacy. Since the block chain is public, everyone can see every transaction that's ever happened, and which wallets hold how much money.

Darkcoin includes a masternode network. These are a sub-set of the nodes on the network that are used to obfuscate the source and destination of a transaction in a similar way to the method the Tor network uses

Pump and dump

Bitcoin's sudden rise in price in 2013 has led many people to believe that similar things will happen for other currencies, and that all they have to do is wait for one to start to rise in price, then buy, and wait to reap the profits.

This has led to the use of pump and dump scams. This is where a group of people artificially inflate the price of a particular cryptocurrency (or other tradable commodity) for a short period of time, then sell their stake while the price is high and leave it to crash.

Inflating the price can be done by pushing out positive news stories that give a false impression of support for the currency, buying up quantities of the currency on exchanges, or almost anything else you can think of.

Before investing in a currency, you should always be aware of the risk of this form of scam. All currencies will have peaks and troughs, and cryptocurrencies are particularly volatile; before investing in a currency, take a look at its history and coverage and decide for yourself whether it seems legitimate.

The next Bitcoins?

- **Litecoin** One of the oldest altcoins, Litecoin was released in 2011. It uses the Scrypt hash and has quite a short block time.
- **Darkcoin** A cryptocurrency with a unique system of masternodes (see main text). Launched in 2014, it's still quite new, but already it's the sixth largest cryptocurrency by market capitalisation.
- **Dogecoin** The logo is of a Shiba Inu dog, which became popular on the Reddit social network. This currency's popularity is almost entirely down to marketing. Users of this currency have raised money to sponsor a Nascar driver, and pay for the Jamaican bobsled team to compete in the Winter Olympics.
- **Ripple** This isn't a cryptocurrency like the ones we've dealt with here because it relies on trust rather than cryptographic proofs. In reality, Ripple is more of a payment system than a currency and isn't easily compared to more common cryptocurrencies.
- **Potcoin** A cryptocurrency set up to support the legal marijuana industry around the world. Some of the proceeds have been used to support the use of the drug for medical uses.

to protect anonymity online. In order to provide some protection against an adversary taking control of a large number of the masternodes, each masternode has to be linked to a wallet with 1,000 Darkcoins.

A useful side effect of the masternode network is that they can be used to guarantee almost instant transactions known as InstantX. Sending using InstantX, a transaction is locked by a group of masternodes until it reaches a sufficient depth. This means that you can have a high degree of security of a transaction with the space of a few seconds.

Trading

It's possible that investing now in the right currency will make you huge sums of money in the future. It's also possible that you will lose your entire investment. Cryptocurrencies aren't a safe way of holding money, but then neither is anything that has such high potential returns. If you want to start trading cryptocurrencies, you'll need two things: a

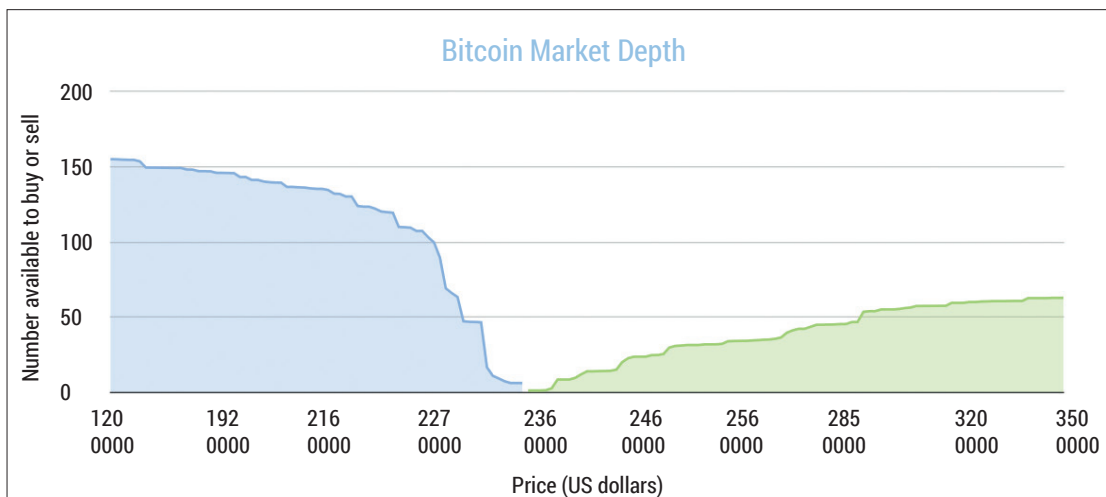


Fig. 1: the price history of Bitcoin in USD from the CEX.io exchange. The red and blue blocks show the range of prices paid for Bitcoin in each time period, while the grey bar graph shows the number of Bitcoins traded in that time period.


thorough understanding of the coins you'll be trading, and an account with an exchange. Trading is all about predicting what will happen, then arranging your currencies to maximise your profit when that happens. You can hold on to currencies for a long time in the hope that they'll continue to rise in value, or you can shuffle money around and try to take advantage of spikes in value.

There are quite a lot of exchanges listed at <https://www.cryptocoincharts.info/markets/info>. It's usually wise to hold some of your coins in a private wallet rather than on an exchange, or spread the risk by having accounts on more than one exchange.

There are two key graphs that you'll see on an exchange that will help you see what's going on: the price history (figure 1), and the market depth, shown in figure two. This is an amalgamation of the various orders out. If you own Bitcoins and want to sell them, you put out a sell order showing the price you're willing to sell them at. If you want to buy them, you do the same but with the price you're willing to pay. The blue line is a cumulative line for the buy orders and the green line is a cumulative line for the sell orders. Where they meet is the current market price for Bitcoins on this exchange. The skill of trading is being able to read these two graphs and deciding what prices to place your orders. 📊



The point at which the blue (buy) and green (sell) lines meet determines Bitcoin's market price.



CREATE YOUR OWN FREE SOFTWARE PROJECT

Got a great idea for an application or game? Not sure how to get started and attract other developers? **Mike Saunders** is your guide.

Free software is tremendously democratic. Anyone with a computer and an internet connection can get involved – there are no barriers of wealth or social status. Being educated in computer science helps, but there are plenty of people working on free software at Red Hat, Canonical and Intel who've never been to university, and who acquired their positions simply by writing great code.

So anyone can contribute to free software, and anyone can start a new project as well. But how do you turn that great idea in your head into a real-life success? The likes of SourceForge and GitHub are littered with now-abandoned projects with barely 50 lines of code, which initially started as grand ideas to create the next killer music player, email client or game. Yes, free software is awesome, but 95% of projects never get off the ground or are abandoned after a few weeks.

Over the next few pages we'll show you how to avoid this, and make sure your project has a proper

chance of being a success. We'll show you how to plan ahead, market your software, pull in new developers and get into the major distros. We're basing this on experience too: this author runs a small operating system project (<http://mikeos.sf.net>) that has had over 60,000 downloads and contributions

from 20+ developers around the globe. Lots has been learnt since the project started in 2006, and we'll share these experiences here.

“Anyone with a computer and an internet connection can get involved with Free Software.”

Naming, hosting, and choosing a licence

It's crucially important to choose a good name from the start. You might be tempted to use a temporary name or some clever geek pun, but if you have to change it later, you'll probably have to use new URLs for your project page and then lose a bunch of hits from web searches. Your name should also be appropriate for the 1.0 release as well. Consider *Minetest*, an open source *Minecraft* clone: it wasn't such a bad name at version 0.0.1 when it was literally just a quick test to see if the developer could hack

together a *Minecraft*-like engine, but now the game is much more complete and the name belies the extent of its features.

Also, avoid using special characters that cause problems with URLs or search engines. At FOSDEM we came across the OP^2 project to build an open source VoIP phone – but just try Googling that name. You'll have absolutely no luck, and if people can't even find your project on the world's biggest search engine, your name means nothing. (We've seen even more horrific examples over the years, including project names containing asterisks and pipes. No, no, no!)

Try to come up with a catchy name, and note that it doesn't have to be directly related to the type of software you're writing. *Firefox* has nothing to do with fire or foxes, but it's a short, unique and slightly odd name that everyone remembers. And again, think of Googlability: don't go too generic. The Gnome 3 team decided to rename its *Epiphany* web browser to simply *Web*, but imagine what that's like for people searching for solutions on Google. "*Epiphany* page load crash" is going to get you much better results than "*Web* page load crash" – so we don't think that was the smartest move by our friends at Gnome.

Finding the right host

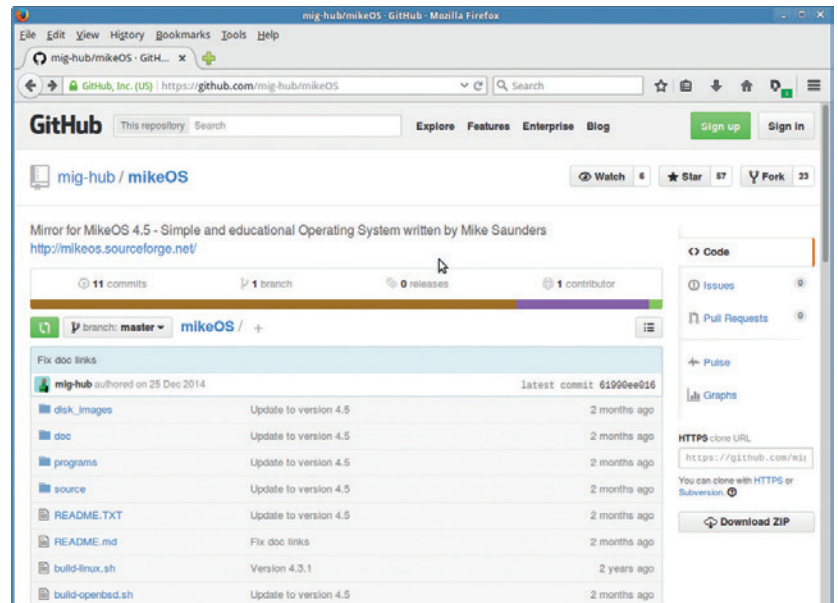
Next, you'll need some kind of web presence for your software. The sheer variety of options on offer can be overwhelming, but there are a handful of websites that we recommend. GitHub (<https://github.com>) is the most popular right now, and lets you create a source code repository that anyone can clone via *Git* and then work on their own branch. GitHub is great for fast-moving projects with lots of contributors, although the services on offer are rather limited – you can't use it to create forums or mailing lists.

SourceForge (www.sourceforge.net), meanwhile, is one of the oldest FOSS project hosts and includes the aforementioned forum and mailing list facilities. You can even administer the website of your project over SSH. SourceForge is a giant site, though, and it can be fiddly to use; we've also seen it become inaccessible

Which language? Which toolkit?

Blimey – those are topics that could fill an entire year's worth of *Linux Voice*. We won't tell you which language, toolkit or framework to use here, as everyone has their personal favourites. C and C++ are important if you want to write Gnome/GTK or KDE applications respectively, and Java is well supported on Linux too. Some Gnome programs are written in Vala, a C#-like language that's loaded with useful features.

If you've never done any programming before, but always wanted to give it a go, we recommend starting with Python. It's easy to read and lets you make command line tools, GUI applications and games. You'll find a friendly tutorial targeted at non-programmers online at <http://tinyurl.com/python3news>, and once you've worked your way through that, you can look at creating desktop tools using *PyGtk*/*PyQt*, or games with *PyGame*. If there's anything you'd like us to run a tutorial about, let us know!



for periods of time. SourceForge got some flak recently for modifying installers of FOSS Windows programs to include (optional) adware, but this hasn't affected Linux users.

If you're passionate about free software principles, want to use the GPL for your licence and plan to avoid all non-free file formats (such as Flash) on your web pages, you could try Savannah (<http://savannah.gnu.org>). Another alternative, especially for those creating software that's tied in with Ubuntu, is Launchpad at www.launchpad.net. Then there's Google Code (<http://code.google.com>) which is fairly limited but very reliable thanks to Google's mighty data centres.

A few notes on creating your web presence: always, always say what your program does right at the top of the page. So many projects have awesome-looking websites that don't actually reveal a single thing about the software in question. Additionally, make sure to include a news section (with dates) on the front page – again, ideally near the top. This way visitors will see that your project is alive and being updated. Try to summarise what's great about your program in a few bullet points, and include screenshots. If you can't make pretty images (eg it's a command line program), try installing a screen recording tool from your distro's repositories such as *RecordItNow* or *SimpleScreenRecorder*, and make a few videos showing off the features.

Choosing a licence

Before you write a single line of code, it's important to settle on a licence for your project. You can't just say that your program is open source or free software – it needs to have some kind of licence attached to it, so that contributors know what they can do with the code. By far the most popular licence in the Linux world is the GNU General Public Licence (GPL), which says that the source code is free for anyone to read, modify and share. But the GPL is more complicated than "just do what you want"; it also enforces these

GitHub is one of the most popular project hosts, and includes an issue tracker for users to report bugs.

GPL
 GPL is the most widely used free software license and has a strong copyleft requirement. When distributing derived works, the source code of the work must be made available under the same license. There are multiple variants of the GPL, each with different requirements.

MIT
 A permissive license that is short and to the point. It lets people do anything with your code with proper attribution and without warranty.

GNU Affero GPL v3.0	GNU GPL v2.0	GNU GPL v3.0
Required <ul style="list-style-type: none"> Disclose Source License and copyright notice State Changes 	Permitted <ul style="list-style-type: none"> Commercial Use Distribution Modification Patent Grant Private Use 	Forbidden <ul style="list-style-type: none"> Hold Liable Sublicensing

The GPL is a good default choice of licence, but see www.choosealicense.com if you want to explore some alternatives.

freedoms on others. So you can't take a GPLed program and make something proprietary out of it – you have to share your modifications under the same terms.

Most of us love this licence, and it has stopped companies from taking the Linux kernel, the GCC compiler, the essential GNU C library (*Glibc*) and other valuable projects and using them in proprietary operating systems. But the GPL isn't everyone's cup of tea: some developers regard it as too restrictive, and prefer the BSD licence. This essentially has the same freedoms to share and modify, but permits code to be used in proprietary software. FreeBSD, for instance, is a BSD-licensed operating system that anyone can download and modify, but there's also a closed-source version included in the firmware of the PlayStation 4.

To use the GNU GPL, see www.gnu.org/licenses/gpl-howto.html. The BSD licence is available at <http://opensource.org/licenses/BSD-2-Clause>, and includes clauses to stop you from being sued if your program doesn't work properly – so if someone uses your code to run a nuclear power station and a catastrophic meltdown occurs, it's not your fault. The

BSD licence also ensures that you get credit for your work when it's used in other projects.

If you're not interested in any restrictions and don't care about being credited as the original author, you can release your code as public domain (www.unlicense.org). Alternatively, you could try the Beerware licence (<http://en.wikipedia.org/wiki/Beerware>) which says: do what you want with the code, but if you find it useful and happen meet me one day, buy me a beer.

Even if you don't care about the licence now, bear in mind your future contributors. Many people are passionate about the GPL and its enforced sharing mechanisms, so it's the best licence to choose if you want to attract the widest pool of developer talent.

Writing the code, and bringing people on board

So many developers make a crucial mistake in the first stages of a project: they start asking around for contributors. This sounds like a great thing to do, and is very tempting – after all, if you tell the world about your awesome idea, you'll soon have a team of 20 hacking away on code, graphics and documentation, won't you?

Well, no. Until you have anything to show, even a 0.1 version, many potential contributors are going to ignore your calls for help. That's not an insult – it's just that there are tens of thousands of projects out there with good intentions, but not a single line of code. If you want people to help you out, you need to show that you're serious about the project, and that you have the knowledge and commitment to do the bulk of the work early on.

Similarly, another problem that can arise in these very early stages is developer spats. If you don't have a basic codebase and roadmap, you could end up with new contributors trying to take the project in many different directions, causing arguments, resignations and (potentially) forks. All this before you've even gotten 0.1 out of the door! It's happened before, and it will happen again...

So it's vitally important to have something to show the world – even if it's a very primitive version of the app. Try to get to version 0.1 or 0.2 on your own. If you have some fantastic ideas for a music player, for instance, but you'll need extra help implementing the advanced features, you should at least get the basics done yourself. Write a simple music player and add menu items or toolbar buttons for the features you plan to add later. This shows potential contributors that you have the knowledge and capability to write an application.

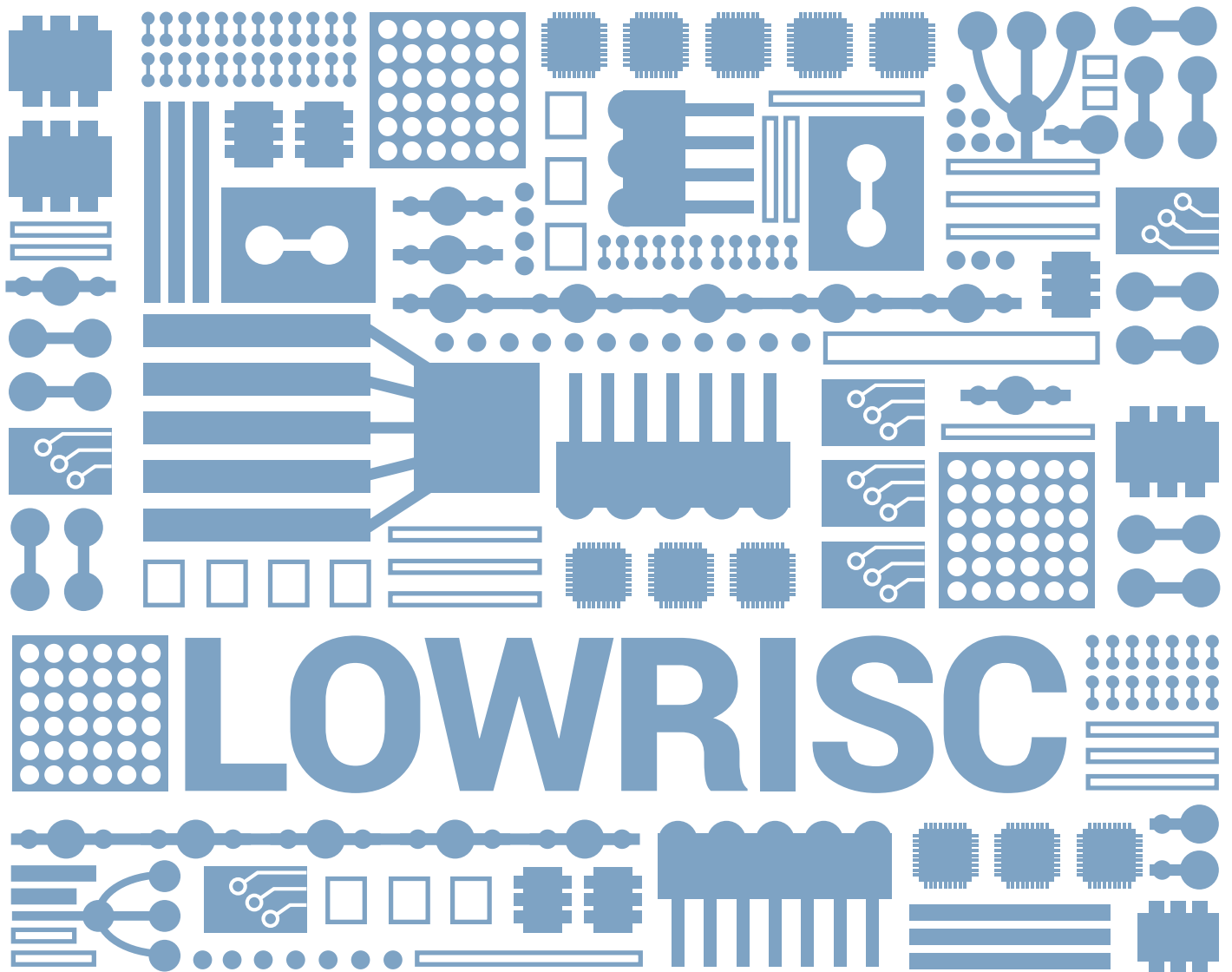
Once you've made a start, try to define a roadmap, at least for the next two or three releases, but ideally up to version 1.0. This helps developers to see your goals more clearly, and prevents your project from languishing in 0.XX versions for many years (like *Inkscape* – a stable, brilliantly useful vector graphics app that's used by professionals, but somehow is still stuck at version number 0.48.2). If some of

Getting into a Linux distribution

Your software will gain the most exposure when it gets into the mainstream Linux distributions. There's no simple way to achieve this, unless you happen to be a distro developer yourself and can simply package up your own work, but there are a few things you can try.

First, look at some related packages and try to find who maintains them. You might see in Arch, for instance, that a certain person is responsible for packaging up various music players. Drop him or her a line, explaining that you're working on a similar program, and it would be great to see it in the distro one day.

Debian has a list of packages called WNPP (Work-Needing and Prospective Packages) at www.debian.org/devel/wnpp. From inside Debian, you can use the `reportbug` tool to submit a request for a new package. Enter `5 – RFP` (request for package) for the report type, and then enter a description according to the template on the web page. This is no guarantee that your program will get into Debian, as there are currently 3,400 requests to package up new programs, but it's worth a shot. Also, if your software finally does get into the Debian repositories, it's likely that it will be picked up by (K/X)Ubuntu and other Debian derivatives.



Discover a project that's hoping to get us one step close to a completely open computer.

It doesn't matter how open or free your software is, the only hardware available today is closed and proprietary. This closed hardware could be used to compromise the freedom of computer use in many ways. Closed hardware can be used to limit what the user runs, the way in which it runs, or what other hardware it runs with. You also can't see how closed hardware works which makes it harder to inspect or improve. Until we have open hardware to go with open software, we'll never have truly open computing. One project hoping to change this is LowRISC.

This project is attempting to design and produce an open system on a chip (SoC) that could be used as the heart of a Linux computer. An SoC is like the motherboard of a traditional computer – it contains the processing core, and much of the associated

circuitry for input and output. The only part of the main system not included on SoCs is the memory.

Having a fully open SoC would put us one step closer to a fully open computer, where the user could inspect the source code for any element of it. Having an open SoC would mean no closed-source blobs to

get it to run. It would mean the possibility of a completely libre computing environment.

In hardware terms, the source code is the design in a hardware description

language (for example Verilog). This compiles to hardware designs in a similar way software source code compiles to machine code. An open chip has to have the code for the hardware description language open so anyone can see it, edit it, and re-distribute it.

The SoC industry is known for its secrecy. Even getting information about how to use particular chips

“Having a fully open SoC would put us one step closer to a fully open computer.”



We spoke to LowRISC co-founder Alex Bradbury after he gave a talk at FOSDEM introducing the project. You can see the slides from that talk at <https://speakerdeck.com/asb/lowrisc-the-path-to-an-open-source-soc>.

can mean signing wide-reaching non-disclosure agreements, so an open alternative here would make it far easier for smaller developers and hobbyists to work with these chips.

Freedom and features

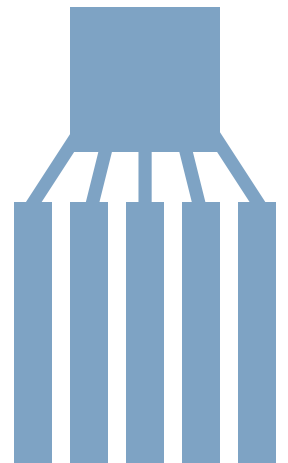
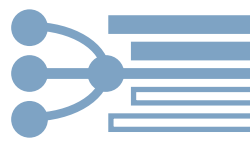
As well as being open, there are a couple of key features that make LowRISC stand out. According to Alex Bradbury, co-founder of the LowRISC project: "I guess the notable features that we're looking at adding are tagged memory support and minion cores. Tagged memory gives you the ability to annotate memory locations to, say, limit access for security purposes, and minion cores are very small, simple RISC-V processor."

These two things make the LowRISC SoC different from other offerings – even commercial chips – but for very different reasons. Tagged memory is basically the ability to mark certain chunks of memory. In the LowRISC solution, this will mean that every 64-bit

word will have an additional two bits of memory associated with it. These two bits can be used to add some context to the word so the processor knows what should be in it. This is most commonly used to enhance security. For example, if an attacker manages to write to memory – such as through a buffer overflow – the processor will be able to see what type of content the memory should have, which will make it harder for the attacker to turn this exploit into code execution.

The minion cores are additional processing cores that sit on the input/output pins. These can be programmed to handle some of the IO activity. For example, if you need a pin to communicate using a particular protocol (such as I2C, which Nick Veitch investigated in LV012 and 013), the minion core can handle the low-level aspects of this communication without taxing the main CPU.

Most SoCs have hardware to handle I2C on a few pins, but the advantage of the minion cores is that you



Open hardware

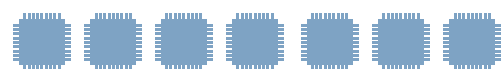
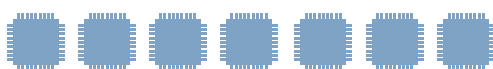
Although open hardware is still a long way behind open source software, there are a growing number of projects that show just how useful it really is. Most of the time though, open hardware only refers to the layout of the circuit, not the actual designs of the chips themselves. For example, Arduino microcontroller boards are perhaps the poster-children of the open hardware world. By releasing all the designs, it makes it far easier to build on them even though the chips themselves are proprietary.

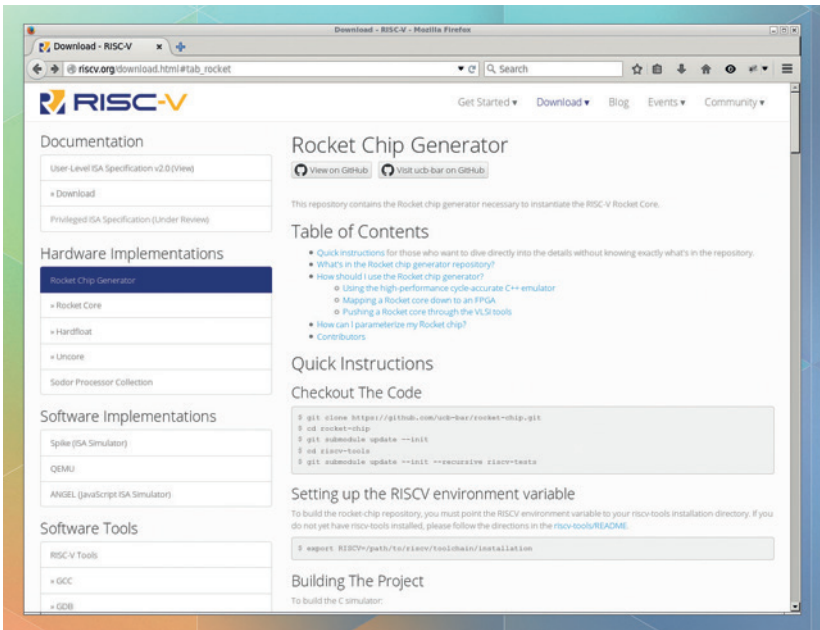
Although this doesn't give the user complete freedom, it does have quite a few advantages. For example, with the Arduino it means that anyone working on a piece of hardware with an Arduino at its heart can easily change the project to incorporate the required features of the Arduino directly into

the circuitry for the hardware rather than including an Arduino as a separate component. This makes it far easier for people to go from prototype to product, even if that product is only something a hobbyist will produce only once.

There are some more open solutions. Sticking with the microcontroller example, you can use the open source ZPU core in an open hardware Papilio FPGA board to create an almost completely open microcontroller that you can use in much the same way as an Arduino. (The FPGA in the Papilio is proprietary, it's just the design loaded onto it that's open.)

Having an open, fabricated SoC will be a huge step forward in the open hardware movement – and indeed the libre computing movement – however it won't be a complete solution until there's open memory, and other associated chips.





You can download code for a RISC-V core (similar to the one at the heart of LowRISC) from the riscv.org website.

can use whichever pins you like for whichever protocol you like, rather than the current situation where protocols are tied to particular pins (as anyone who's used the Raspberry Pi GPIO pins will know).

In many ways, the situation with minion cores is a little like having Arduino microcontrollers sitting between the CPU and the IO pins. For anyone building custom hardware, this could be very useful.

Although LowRISC is trying to make a fully open SoC, it won't be able to completely avoid proprietary code from the first version. Alex Bradbury told us: "As for all this open source stuff, there are a whole bunch of lines that you can choose to draw. The lowest aim is for everything that you would implement in a hardware description language (like Verilog) to be fully open – so all the digital logic is fully open. [However,] it might be that in the initial case, we need to take on some closed source intellectual property for some IO controllers, because often the physical interface is very tightly integrated and tightly tied to the controller. It might just be too much engineering work for the first chip."

While we do feel that it's a bit of a shame that the initial SoC won't be completely open, we understand the need to be pragmatic when bringing something as complicated as an SoC to market. Since the RAM will be in a separate chip, this closed source code will just be dealing with the interaction with that memory. This is something, Bradbury explained, that the team would like to open in future revisions.

LowRISC isn't the only project trying to create open source SoCs. OpenRISC has been around a lot longer and has several designs that can be implemented on Field Programmable Gate Arrays (FPGAs). These are chips that contain lots of logic circuits and a mechanism to connect these circuits in different

ways. This means you can download different hardware designs onto the chip and run it. It's a cheap way of trying out different designs.

There are some important differences between LowRISC and OpenRISC. "I suppose the comparison between first the OpenRISC versus RISC-V, the difference would be that RISC-V is a clean slate, 64-bit ready rocket architecture with a very minimal instruction set whereas OpenRISC perhaps made the mistake of throwing in too many instructions into their basic architecture," says Bradbury. "It's taken them a long time to get a 64-bit version and adding atomic support, that's just about happening now. I think there are very friendly people in the OpenRISC community and most of them kind of see that if they did a clean 64-bit OpenRISC, it would probably look something like RISC-V. Indeed there is a fair share of lineage."

Performance

The crunch question for any new computer is what speed will it run at: "The aim that we discussed for our first meeting is to run Linux well. This is what we're looking at: dual- or quad-core running at 1–1.5 gigahertz, the exact clock speed will depend on if the production process ends up at 40 nanometers or 20 nanometers." The process size refers to the smallest

imprints that can be made on the silicon. The smaller process would mean a faster chip.

Running Linux "well" is a – perhaps intentionally – vague target. For many users,

the limiting factor of the first LowRISC SoC will be the lack of a GPU. This means it will struggle with most desktop uses. However, there are still plenty of applications where this isn't such a problem: "There are a number of people who are particularly interested

"Already open source contributors are making key additions to the project."

Genesis How the project got started

Although LowRISC will be producing and selling silicon chips, the aim behind the project is to make open hardware, not money. The organisation is registered as a community interest (i.e. not-for-profit) company.

Alex Bradbury says: "We started around last summer when [Dreamworks'] Gavin Ferris got in touch with Rob Mullins [of Cambridge University] and myself. We started discussing what we could do to make open source hardware happen, what the opportunities were. So we started looking around to see what was going on in that space. There are existing things like the OpenRISC project, which has its own open source architecture. We found Krste [Asanović]'s team at Berkeley, who Rob has worked with before, and that very fortunately, the chips with RISC-V work and have this RISC-V Rocket core."

Gavin, Rob and Alex form the main part of the team, but there is also a technical advisory board made up of some key people in the open hardware world including Julius Baxter from OpenRISC and Bunnie Huang, open hardware advocate and hacker extraordinaire who crowdfunded an open hardware laptop – www.bunniestudios.com/blog/?p=3657.

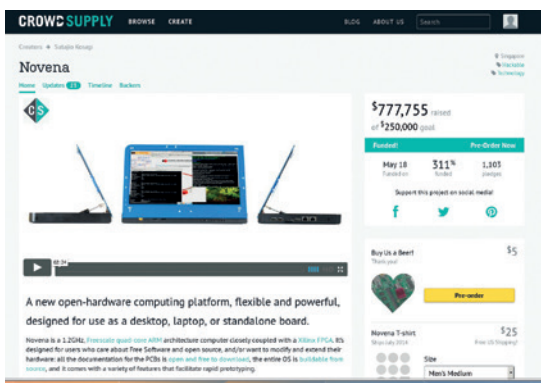
in security-type applications [because of tagged memory]. Given that we ship without a GPU, an obvious use case would be a router. Right now it is a real problem that you buy your off the shelf hardware and it is running some firmware that you can not change that was written six months ago already has a local root vulnerability you cannot patch around."

Obviously routers are not the only hardware that doesn't need graphics but that does need good security, so we expect to see a lot of projects pop up once the hardware is available.

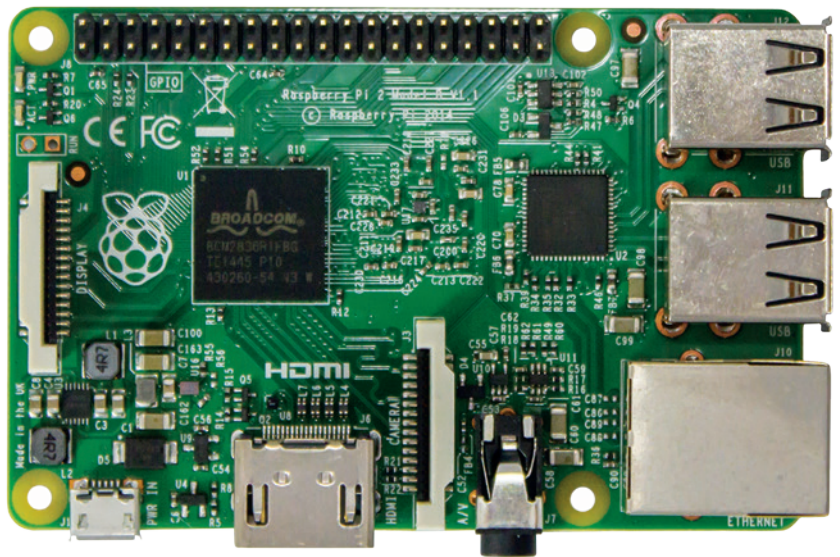
Community

If this sort of thing sounds interesting to you, then the LowRISC team are keen to find people to help out: "We're very welcome to contributors, we want people to get involved," says Bradbury. "Right now, as with many open source projects, the documentation isn't quite where we want it to be. But in the future, if somebody here is interested in learning more about hardware design and about how you can apply some skills that you might already have in software to the hardware world, then we want to be the source of that. There are all sorts of levels you can get involved, from people who have some basic software knowledge and want to work on software reporting to people who want to work on documentation."

This isn't just an abstract hope for the future. Already open source contributors are making key additions to the project: "One of the pleasing things that we found was that when we first announced what we are doing with the project – which was sometime last year – it coincided with some press from the Berkeley team [who are creating the RISC-V core]. We were a little bit apprehensive about doing that because we did not have an FPGA thing ready. We thought we should wait a bit longer, but what we found is that we have been able to move at a much faster rate than we would have done otherwise due to people contacting us and offering their help to bring in some ideas. Right now there are a number of people who have been very helpful in terms of making suggestions about design decisions."



The Novena open hardware laptop, created by LowRISC advisor Andrew 'Bunnie' Huang, raised over three quarters of a million dollars (three times the goal) in 2014, showing just how much desire there is for open hardware.



Bradbury continues: "We started seeing more design discussions go on the normal mailing list about how things such as the link between the application cores and the minion cores should work."

The biggest difference between an open hardware project and an open source software project is that it's effectively free to compile software, while it can be very expensive to make new hardware. Alex told us how LowRISC has overcome this hurdle: "We are very lucky to have some initial funding from a private backer, which is about enough to get us off the ground and have some people working on it through to the initial test. We have access to loans... and we're also applying for more traditional research funding and for the research aspects of the chip."

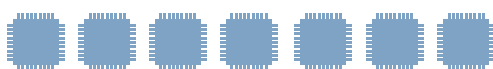
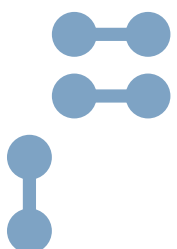
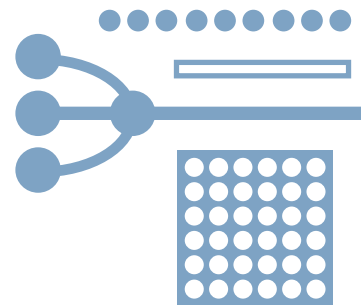
The start of something big

"When it comes to producing the final board, we may well end up doing a crowdfunding campaign if that seems a sensible way to go, but we decided it is not something we want to do until we are basically almost there and it's just the case of needing the money. Right now there are too many unknowns and people wouldn't know exactly what they were buying, so I do not think it would be quite right to start at this stage."

With the money in place, the LowRISC team hope to have something for us soon. The initial test run of chips should run toward the end of 2015 or at the start of 2016. This won't be a full-scale production run, but will produce enough for there to be some available for people in the community to start developing software ahead of a main run later. You don't have to wait until then to start playing with LowRISC though. The team are putting together a version that you can load onto an FPGA board: "If you go to the RISC-V website you can download Rocket Core, which will run on a Zynq FPGA. We have an FPGA-ready version with tagged memory, and then some time this summer some of the minion cores as well."

You can stay up to date with the project, or get involved with the development, at www.lowrisc.org.

Although the projects have different goals, LowRISC shares some heritage with the Raspberry Pi Foundation.



UNPROTECTED COMMUNICATION



Mass surveillance violates our fundamental rights and is a menace to the freedom of speech! But: **we can defend ourselves.**

The password that protects your email is not sufficient to protect your mails against the mass surveillance technologies used by secret services.

Each email sent over the internet passes through many computer systems on the way to its destination. Secret services and surveillance agencies take advantage of this to read millions and millions of emails each day.

Even if you think you have nothing to hide: Everyone else that you communicate with via unprotected emails is being exposed as well.

Take back your privacy by using GnuPG! It encrypts your emails before they are sent, so only the recipients of your choice can read them.

GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

About GnuPG

Thousands of people already use GnuPG, for professional and private use. Come and join us! Each person makes our community stronger and proves that we are ready to fight back.

Whenever an email that is encrypted with GnuPG is intercepted or ends up in the wrong hands, it is useless: Without the appropriate private key it cannot be read by anyone. But, for the intended re-

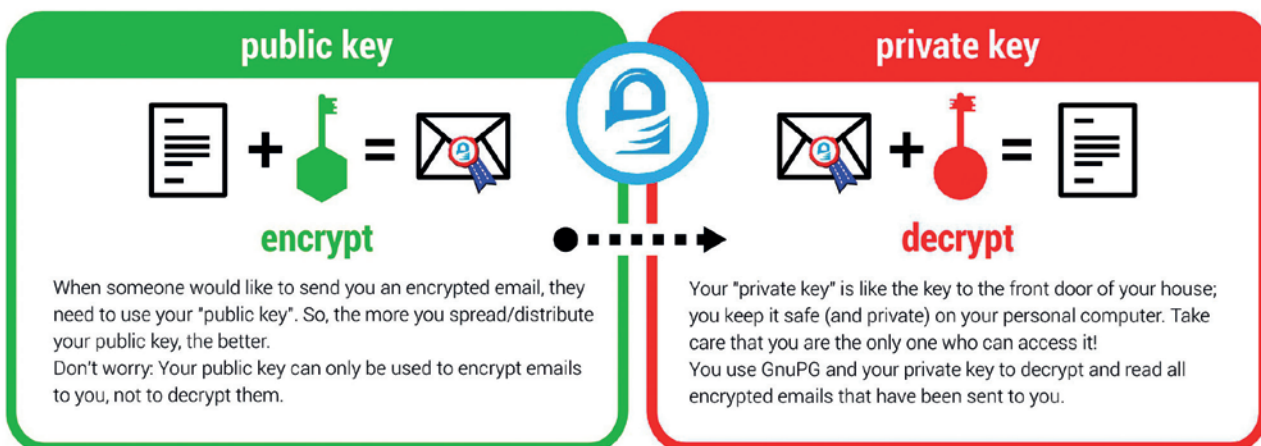
ipient - and only for her - it opens like a totally normal email.

Sender and recipient are both safer now. Even if some of your emails contain no private information, consistent use of encryption protects us all from unjustified mass surveillance.

What makes GnuPG secure?

GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.

If no one is allowed to see the source code of a program, nobody can be sure that it does not contain undesirable spy programs - so-called "backdoors". If software does not reveal how it works, we can merely trust it blindly.



GPG-ENCRYPTED COMMUNICATION



In contrast, a fundamental condition of Free Software is to publish its source code: Free Software allows and supports independent checking and public review of the applied source code by everyone. Given this transparency, backdoors can be detected and removed.

Most Free Software lies in the hands of a community that works together to build secure software for everyone. If you want to protect yourself from surveillance you can only trust Free Software.



Practical advice

The technology behind GnuPG provides first-class protection. To ensure that your encrypted communication is not compromised for other reasons, use a strong passphrase and backup your private key. Encrypt as much as you can! By doing so, you prevent others from realising when and with whom you exchange sensitive information. Thus, the more often you encrypt your messages, the less suspicious encrypted messages will be. Be aware that the subject is transmitted unencrypted!

You can find a simple tutorial for email self-defense with GnuPG encryption here:
EmailSelfDefense.FSF.org

Watch out for so-called "Cryptoparties" in your area! These are events where you can meet people that would be happy to help you in setting up and using GnuPG as well as other encryption tools at no charge.

About the fsfe

This article was created by the Free Software Foundation Europe (FSFE), a non-profit organisation dedicated to empower people in Europe in their use of technology by promoting software freedom.

Access to software determines how we can take part in our society. Therefore, FSFE strives for fair access and participation for everyone in the information age by fighting for digital freedom. Nobody should ever be forced to use software that does not grant the freedoms to use, study, share and improve the software. We need the right to shape technology to fit our needs.

The work of FSFE is backed by a community of people committed to these goals. If you would like to join us and/or help us to reach our goals, there are many ways to contribute. No matter what your background is. You can learn more about this under: fsfe.org/contribute

Donations are critical for us to continue our work and to guarantee our independence. You can sup-

port our work best by becoming a sustaining member of the FSFE, a "Fellow". By doing so, you directly help us to continue the fight for Free Software wherever needed: fsfe.org/join

What is Free Software?

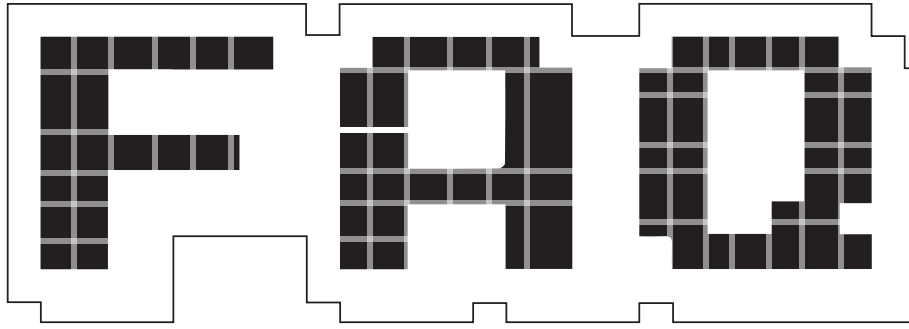
Free Software can be used by everyone for any purpose. That includes free copying, reading the source code and the possibility to improve or adapt it to your own needs (the so-called "four freedoms").

Even if you "only want to use" the program, you still benefit from these freedoms. Because they guarantee that Free Software remains in the hands of our society and that its further development is not controlled by the interests of private companies or governments.

Find out more about this and how Free Software can lead us into a Free Society: fsfe.org/freesoftware

If you like to help spreading the word, you can order this and other leaflets under: fsfe.org/promo

fellowship
of fsfe



HTTP/2

Graham Morrison reviews the sequel to the most common acronym on the internet.

GRAHAM MORRISON

Q Haven't we seen the acronym HTTP before?

A You've probably seen those four humble letters so many times that you've become completely desensitised to their appearance. It's the modern fashion to remove them from URLs because they're everywhere, but they do perform a vital function. These are the letters that tell your web browser what type of resource is at the end of that link, and for the vast majority of connections, the resource at the end of that link is a web page.

Q You mean there are other kinds of resources?

A Yes, but not so many any more. At least not ones that a web browser knows how to deal with. Web browsers used to be able to interpret all kinds of different resources. FTP is still common, for instance and so is 'mailto'. In the mid 90s there used to be more as the browser was designed to aggregate

"HTTP/2.0 does lots of sensible things designed to improve transfer speed."

all kinds of online content. Many, such as the Gopher protocol, can still be added via plugins, but these days it's all about the web, and that means HTTP.

Q What do the letters HTTP represent?

A What we've been calling 'resources' are actually protocols that grab stuff. A protocol is a definition of how those resources should be formatted and transferred. The 'P' of HTTP is 'protocol', while the HTT bit is Hypertext Transfer.

Q Hypertext? From the early 90s?

A Yes, the very same. It's a word that's fallen out of fashion, but its meaning is fundamental to how the world wide web works. The hype in hypertext is derived from the original Greek meaning of 'over', or 'beyond'. Or within a text file, it's the link to another resource 'beyond' the limits of the current file or location. This linking is what makes the world wide web the world wide web. The Hypertext Markup Language (HTML) is the syntax and formalisation of that linking with the text that surrounds it.

Q So HTTP is the protocol used to send HTML?

A Fundamentally, yes. At least in the beginning. The simplest

implementation of HTTP would have only a single command, GET, which would request an HTML file from a server. That HTML was nearly always a static file formatted with the correct markup. Markup refers to the elements within an HTML file that tell the browser how to format the text, such as `<h1>heading</h1>`, for a title or a heading. There are many elements and rules and we've all created files like this at one time or another.

Back at the dawn of the web, everything was made up of static sites like this, simply delivering a formatted text document to your browser. But as the web has evolved, HTML has become dynamic, created by whatever is running on the web server and code running in your browser. *WordPress*, for example, will take the posts you insert into a database, blend them with your themes and comments, and deliver the final output to someone's web browser, whether that's a phone or a laptop. Tim Berners-Lee is credited as the first to implement HTTP and HTML and made the first transfer back in 1991.

Q Is that how browser games work?

A Not usually. Most of these are written in JavaScript, a scripting language that's sent as part of a page and executed within your browser, but there are lots of other similar

technologies. They can even be sent through a connection after the original request as part of the same session. Allowing a single session like this was one of the new features for HTTP/1.1.

Q How does all this fit into what the internet is?

A It's easy to get into a technical discussion about this, and in particular, take a deep dive into network layers. Briefly, HTTP operates at the top in a layer known as the 'application layer'. Your web browser asks for a page and the server at the other end replies by sending it. It shares this space with many other protocols – IMAP for email, or SSH for a remote shell, for example.

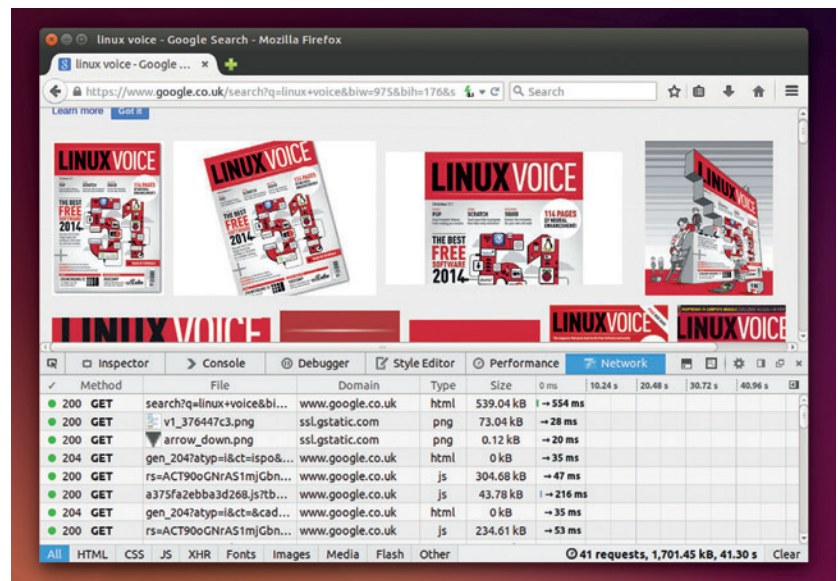
Even if you have no formal computing knowledge, these protocols will be familiar precisely because they're in the Application Layer, the layer closest to the user. If you look at the IRC protocol, for instance, you'll see that it's very simply constructed. Communication is really just a series of text messages that you can recreate manually using something like Telnet. You don't have to worry about how your messages are encoded, or how they get from your machine to the server. This is handled by the layers beneath: Transport (TCP), Internet (IP) and link (Ethernet).

Q If all HTTP is doing is enabling a client to ask for data from a server, why does it need upgrading?

A The HTTP that most of us use is version 1.1. This has been around since 1999, when Google had just eight employees and was moving from its garage office to its first real office. Just as 1.1 added features that were becoming necessary as the web grew in importance, so too does HTTP/2.0. It's remarkable that the old version has lasted this long, considering what's happened in the intervening 16 years.

Q But what does the new version do that's so important?

A Put simply, speed. We now know so much more about how we use the web and what the user and the web designer are trying to achieve. HTTP/2.0 does lots of sensible things designed to improve transfer speed and efficiency between the client and the server. HTTP is no longer going to be



There's an add-on for *Firefox* that will show you when you're using an HTTP/2 or a SPDY connection (it's the tiny green symbol in the location field).

text-based, but use binary instead, for example. It will be the same content, encoded for efficiency. HTTP/2 uses gzip or DEFLATE compression and multiplexes transfers within a single connection. TLS security, which you currently use with HTTPS connections to your bank, are also an intrinsic part of this 2.0 specification, making HTTP connections implicitly secure.

Q Are there any other advantages?

A Lots! It's a free upgrade in that the new version won't require you to change anything, or for developers to change their APIs. The new version will just work. As fewer connections are needed, the load on your server will also be less. There's more intelligent cache control, and the server can push data it thinks the client will need without being asked, which should improve response times. Plus, encryption becomes a first-class citizen.

Q Has Tim Berners-Lee had a hand in this upgrade?


A Not specifically. HTTP/2 was approved in the middle of February 2015 by the Internet Engineering Steering Group. HTTP is so fundamental that no decisions are ever made quickly, and decisions like this are only made after a long and peer-reviewed appraisal process. For HTTP/2, that means 200 design issues, 17 drafts and 30 implementations.

HTTP/2 is based on a technology originally developed by Google, called SPDY. SPDY modifies the HTTP transfer in similar ways, only hidden behind compatible clients and servers. Google was well placed to deliver a specification like this, considering the free bandwidth upgrade it would receive from any efficiencies, and SPDY had already been adopted by all the main browsers as an addendum to the old specification. Google is now going to withdraw SPDY from its own products to help get HTTP/2 adopted as quickly as possible.

Q Do I need to change my browser to use this?

A *Firefox* has HTTP/2 enabled from version 36 onward, and *Chrome* supports HTTP/2 but it isn't enabled by default. The version of *Internet Explorer* bundled with the latest Windows 10 beta also support the standard. Each of these browsers only supports the encrypted (TLS) version of the protocol. *Safari* supports SPDY and is likely to adopt the changes necessary to add HTTP/2 support, so there should be good cross-platform adoption.

Q Where can I find out more?

A The implementation lives on GitHub: <https://http2.github.io>, but you can find clearer information on the HTTP Working Group's own web portal: <https://httpwg.github.io>. 

A MAN WITH HIS FINGERS IN MANY MILLIONS OF PIES

Graham Morrison and Ben Everard meet the man responsible for more 'pie' puns than anyone else in the world – Eben Upton.

At the time of writing, over five million Raspberry Pis have been sold. That's the same as the number of ZX Spectrums sold in the 80s. And like the Spectrum, the Pi is likely to have a far-reaching legacy, helping the next generation of games designers and computer scientists find their feet.

Countless numbers of people have helped make this happen, but

Eben Upton has been there from the beginning. He's the founder and the CEO of the Raspberry Pi Foundation, and he's still shaping every aspect of the Raspberry Pi, from its hardware to the software. We met Eben shortly before the launch of the model 2. He told us about the effort they've put into making the Pi better and how a chance conversation with the boss of Google shaped the Pi's future.

LV When did you start work on the recently launched Model 2?

EU: When we launched the B+, we already had Pi 2 silicon – this was the start of May last year (2014). And so the last bits of B+ design were done in the knowledge of the pin out on the [Broadcom] BCM2836 [the quad-core ARMv7 CPU on the Raspberry Pi 2]. It was designed so that, probably, we'd be able to squeeze Pi 2 into the same footprint.

LV Was that part of the motivation behind the B+ redesign?

EU: Not really. It was something we'd always wanted to do.

LV With only a nine months gap for sales of the then new B+?

EU: Yeah. B+ was a bit late. We wanted to do the rev. 3, which became B+. Ideally, it would have been good to do that six months to a year earlier than we did, but we just didn't have any engineering time, partly because we were involved in some silicon work for BCM2836. We had limited resources and they didn't quite stack up in order to get B+ out. Any less time on the market may have been a bit embarrassing, but B+ has had a solid six and a half months, which is alright.

LV How do you anticipate demand for the Model 2?

EU: Today is 20 January (the model 2 went on sale 14 days later), there are 10–20,000 in existence today. Sony will be building 20,000 a day at the end of the month, so there will be between 100 and 200 thousand on 2 February.

LV That will be in the past by the time this comes out, so you can say, "It's gone brilliantly!"

EU: It's gone brilliantly! Nothing went wrong. Everything's great and everyone was very happy [much laughter]. We'll see. I think people will be excited.

We've had sequential best months over the last few months. All of our last three months have been in the top five months we've ever had, in terms of volume. It feels like we're going out on a high. It's still a great platform. It's still the best out there and we're just making it better.

LV Do you have any indication of where those 200,000 will go?

EU: Less than you'd think really. We know that maybe a third are going to industrial customers now – these kind of big batches that fall into a hole and disappear. People are using them as industrial controllers, people are



prototyping other products. That happens a lot. Then the rest, it's a split between the hobbyists and education. It's pretty close, I think, to a third, a third and a third: industrial, hobbyist then education. You've got people who are buying them for themselves, to use as their own media centre, in the hobbyist sector, and then you've got the industrial sector who are buying and integrating them in a sense and selling them to people.

LV Has there been much traction with the Compute module?

EU: We've sold a lot of dev kits but not a lot of Compute modules, which is probably what we expected. I think the designing cycles proved to be longer than we were expecting, so we haven't had any volume Compute module customers. With the exception of a few things like Slice (a media player using the Compute module), where we've had



a few thousand. But we haven't seen those 100,000 orders yet. I think it just takes a while to design anything and you could only buy those in June 2014.

“I do word processing on my BBC Micro on my desk... I hit the button and it's on.”

LV When you say you think it's about one third education, is that almost exclusively UK?

EU: No, we've got a lot of stuff in North America, quite a lot of stuff in Germany. The UK is leading the way – we are the first large-population country to have a decent computing curriculum (Estonia beat us). But a lot of the sales aren't curriculum for the class – parents buy them for their kids, clubs buying them, grandparents, that sort of thing.

LV And the model 2 still has the Ethernet connection going through the USB?

EU: We've taken the long road to get here. People have been shouting at us for two years “Why don't you go get one of those ARMv7 (non-Broadcom) chips you can get for £5 in China?” The reason is that I'm not a great fan of them. And also it would just break compatibility with the Pi, and it wouldn't be a Raspberry Pi: it would be another computer, with a Raspberry Pi logo on. We've taken this long road of getting an SoC which has better ARM performance but is otherwise identical to any Pi device, in order to bring the community with us. So we avoid ending up either supporting two platforms or, more likely given the number of people we have, ditching support for the old platform. So we are going to keep supporting Raspberry Pi 1 for at least a couple of years. And we're likely to keep

on using Raspbian instead of upstream moving to regular ARM Debian. We're going to benchmark it and find out. What I suspect we'll find is there'll be a few libraries that pose big performance improvements, and we'll swap those out dynamically.

LV Is that for the original Pi as well? We're assuming the video decoding stuff in the original Pi was more suitable for a set-top box or something you'd put under a TV, whereas ARM since then has been used in so many smartphones.

EU: The SoC has been used in mobiles, set-top boxes and obviously Raspberry Pi, so it's a versatile little beastie.

LV Have you been as involved in the Raspberry Pi 2 design as the original?

EU: Yeah. The hardware design for this was done by James Adams, who's our designer. He designed the B+ and the A+. The design for the original hardware was done by Pete Lomas. So it's been good to be involved, good fun.

LV Is this the kind of release cadence we can expect to see from the Foundation in future?

EU: What, three years? Let's see, where does that take us to? 2018. Yes, I do think we'll probably want something else by 2018. Yeah, I think that's about right. That gives us three years at \$35. That's the lovely thing about this - it gets us into the PC world. It gets us into the entry level PC. It's the kind of device you could give to your gran. We have people here that are using them as their second machine at home.

LV Are there any new use cases that you think the Pi 2 will fit into with the extra power?

EU: The desktop is a good use case, running multi-threaded apps.

LV You said this Pi 2 has been in development for two years. At what point during Pi 1 development did you realise you needed to start designing Pi 2?

EU: It was about six months in that we became aware. We sort of stumbled on this thing, so towards the end of 2012 we were like, “OK, how do we get a platform going now?”

LV Or what the environment might be like in two years time?

EU: Yeah, or what will be relevant to us, what sort of price trajectories of the components would be and seeing what we could fit in – quite early on. I had a really good conversation with Eric Schmidt [executive chairman of Google] in January – so, exactly two years ago. We'd had all these ideas about doing a more expensive model, because there were lots of small board computers that cost around \$50, \$60, \$70, right?

We were really starting to see things at \$60–70, because now there was a profusion of \$50–70 machines, and I think we can take credit for catalysing that development. We were thinking we could do something really great at \$60 and I happened to mention it to Eric. He said, "So what are you going to do next?" I told Eric we'd been looking at making a more expensive model.

Eric said "Don't be an idiot!" or words to that effect. "That's completely insane – see if you can make a free computer." Because obviously I'm patting myself on the back, thinking, "Yeah, you know, I made this \$25 computer - then he resoundingly refused to pat me on the back and slagged me off for it being too expensive."

LV Did you understand what he said instantly?

EU: Yes. I went back to the office and I cancelled every investigation into a faster machine. That afternoon. Because it was the right thing to do and as soon as someone says it to you, particularly someone like that who

you're not going to question really. You kind of think, this guy obviously knows what he's doing, he's got the biggest brain on the planet. I went back to the office and cancelled everything. Every investigation we had into a higher performance, incompatible, more expensive device.

So the two outcomes of that one conversation are this thing [Eben picks up the Raspberry Pi 2] and the A+. A+ is the 'can we make it cheaper', 'can we sacrifice anything', though of course we're not really sacrificing anything. I think without that conversation, the A+ might have been a \$25 machine with half a gig of RAM. We might have ended up there instead. So the sacrifice in some sense is a bit more RAM – that was one outcome of that conversation. The other was the Pi v2, and they both took about two years.

LV Has it surprised you that no one has come along and tried to create something cheaper?

EU: Yeah, it is surprising, I think. It's surprising to me but it is tough to do. It's difficult and it's hard to do at scale. I think you can make these things for this sort of price – and you've seen one or two people manage while making sacrifices, often using a less good SoC. But Sony will be building 20,000 units a day, at peak, of the Model 2, and it's really hard to do that.

So it's that combination of cheapness and availability. There have always been small working computers. But often they weren't available. They were a lovely idea but you just couldn't buy

them. They were not in stock – you had to wait for the next delivery, or for the next batch of 1,000 units to be made.

You can build a business around [Raspberry Pi] because you can depend on them existing – like with the Model B, we still build Model Bs – these guys have sold tens of thousands, probably, we've built new (old) Model Bs after the launch of the new Model B+, not just selling our old stock, but building tens of thousands of Model Bs to supply industrial customers who built a business around the Model B and they haven't been able to transition to the B+ yet. And I think the same thing may happen with the B+ – we've sold a couple of hundred thousand B+s.

LV Despite Moore's law, our computers don't feel any faster than they did 20 years ago. Even an old Raspberry Pi has millions of cycles per second to use.

"We've got a shopping list for ARM7. We know what attention to detail actually looks like."

EU: It's got a lot to do with Parkinson's law too – work expands to the time available. So this is kind of the CPU version of Parkinson's law. One of the things we've done with the Pi is refuse to accept that 700 million cycles-per-second in the processor is slow. I just won't accept it. And people kept saying, "It's such a slow processor." and I'm saying "It can do 700 million things a second! Your high-definition screen has only got two million pixels. You can do 350 things to each pixel on your screen every second. How is that slow?" I don't believe it, and [the Pi] has even got vector operations, and the FP [floating point processor] has even got multimedia accelerations, MMX-like operations. So I just refuse to believe that they're slow.

We've spent so much money on open source software. We've spent so much money on the *WebKit* browser engine and *Libav* [for multimedia playback] and *Squeak* [an open implementation of the Smalltalk programming language] and *PyPi* [the Python Package Index] and *KICAD* [software for designing electronic circuits], *LibreOffice*, even. You look at



The Raspberry Pi isn't just about the hardware – the chap on the right here is Ben Nuttall, part of the Foundation's education team.



By the time you read this the Pi's sales figures will probably be some way in excess of the five million milestone.

these packages and you think well, I just refuse to believe that this should be slow. And you go through and you find out why. This is the thing no one ever does. Everyone has just been mesmerised. You could just sit on your arse and in two years time, everything will be twice as fast, and that's coming to an end. So we're kind of like the future. This increase in performance is not available to us any more.

LV But also, it hasn't succeeded. Our desktops and our laptops are still slow.

EU: I've got a BBC Micro on my desk because I do word processing on my BBC Micro. I hit the button and it's on. If I set the jumpers right I can even configure it to boot into a word processor in one second. Obviously we've gained a lot. But from a strict usability point of view it's really surprising how much everything hasn't got any better.

It's one of the reasons why I like to ship RiscOS, because although RiscOS isn't a modern operating system and it lacks lots of things, it does show you what you get if you took an OS from the mid-90s and don't add any crap to it in the form of features people don't need. And the answer is blindingly fast.

So what we've done with Pi 1, we've gone through everything. We've found out why it's slow. We found out why the web browser was slow. And then we went through and we laboriously turned on features, because actually *WebKit* has got lots of great features, because it will run on mobile phones, but none of those features are turned on in any of the desktop versions. So we went through and we turned all of those on, plumbed in the video accelerators. We just did that work.

You've got *WebKit* and *Cairo*; we went through and we found every pixmap call they made and then we made sure they used the fastpath mode and made sure that all of them hit fastpaths rather than the generic and very slow CPU compositor format. Same with *Libav* – all your codecs and transport streams, remapped.

We've put two man years into Squeak and Scratch. Two man years on *Pixman* and *Libav*. Four man years on the web browser.

LV That's a huge amount of effort to put in when you could have just doubled the CPU speed!

EU: Absolutely! There's a factor of two improvement from all the optimisations we've made. It's worth going back on

the Pi and running an old image – go and run an image from March 2012. It's amazing, the performance difference. It's amazing how much of a difference non-optimised memory copy makes.

LV And you don't have to abandon all that optimisation work with the new model?

EU: You don't abandon that. We would have had to abandon it if we'd moved to a different SoC. So you've got that factor of two in the attention to detail. Some of that stuff will be ARMv6 specific, so we've got to go back and do that for ARMv7, so there is more work to do, but the nice thing having done it with ARMv6 is that we've now got a shopping list.

We know what attention to detail actually looks like – it looks like writing all of these functions and we'll just go back and say, for each of those functions is there already an ARMv7 version? If there's already an ARMv7 version, do we like it? If we don't like it let's rewrite it. If there isn't one, can we use the ARMv6 function that we already wrote? If we can't use it, then use the ARMv7 version. We now have a shopping list, a kind of handle-turning exercise that will make this platform better over the next six months. **LV**



LISTEN TO THE PODCAST

LINUX VOICE

WWW.LINUXVOICE.COM



BUY LINUXVOICE MUGS AND T-SHIRTS!



shop.linuxvoice.com



Andrew Gregory

Cat eggs in the flower patch? Time to build the anti-feline garden intrusion detector.

Elementary OS got itself in a bit of a pickle earlier this year by asking its users for money. Except, it didn't. There's nothing wrong with soliciting donations, selling products or services or raising cash via the world of crowdfunding, as the Krita editor has done to stunning effect (see page 53 for more on why we've fallen in love with it). No, Elementary got itself into a pickle with the Gerald Ratner-esque assertion that users who download its Ubuntu-spinoff distribution are "cheating the system" when they download the distribution without paying for it.

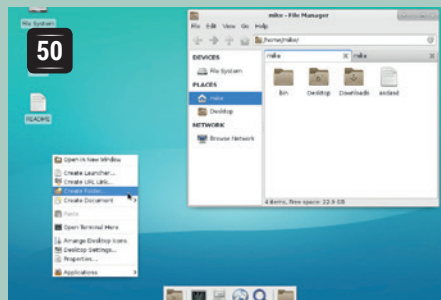
Never mind that Elementary is a spinoff from Ubuntu (which itself is derived from Debian), which is doing very well for itself without feeling the need to insult its users. Never mind the silly language. The fact is that, if there were an up-front price tag on Elementary OS, it would lose almost all of its users.

In a world where there are so many almost perfect substitutes for Elementary (Xubuntu, Lubuntu, Debian, Mageia *et al*), all free of charge, any other desktop distro will have a hard time convincing anyone to pay for it. Charge for something else – upgrades, codecs, early access to features – but expecting people to pay for what they can for free elsewhere is silly.

andrew@linuxvoice.com

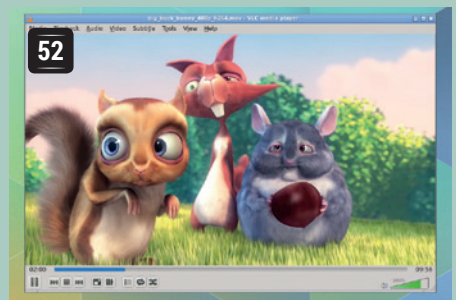
The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

On test this issue...



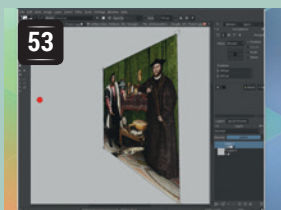
Xfce 4.12

Good-looking, lean, fast, full of useful features and refreshingly old-fashioned, **Mike Saunders** is the perfect man to test the latest *Xfce* desktop.



VLC 2.2

Ben Everard explores every last corner of the epic *VLC* media player, an application that can open just about any video format ever created.



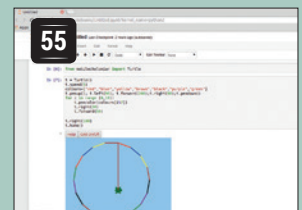
Krita 2.9

Native CMYK support, a modern interface and a load of new tools make *Krita* **Graham Morrison's** favourite image editor.



Inkscape 0.91

Don't let the lowly version number fool you – this vector graphics tool is one of the best on any platform, says **Graham Morrison**.



IPython 3.0

This interactive programming environment provides a whole lot of fun for the mathematically minded **Ben Everard**.

BOOKS AND GROUP TEST

Now that the internet of things is upon us, and many households will have several devices at home that can all attach to the internet, it makes sense to have one place to store media in the house that all these devices can access, rather than having them scattered piecemeal on several devices. That's the whole point of a NAS box (Network Attached Storage); there are many ways to run a NAS depending on your needs, and our Group Test this issue is dedicated to helping you find the best one for you. Also – there's a new book by Bruce Schneier!



Xfce 4.12

It has taken almost three years, but a shiny new version of this GTK-based desktop is here. **Mike Saunders** investigates.

DATA

Web
www.xfce.org
Developer
Xfce team
Licence
GPL, LGPL and BSD licences

Xfce's position in the Linux desktop ecosystem has changed considerably in recent years. At one time it was widely regarded as a lightweight alternative to Gnome, but when that desktop had a massive (and controversial) redesign in Gnome 3, Xfce became the "classic" alternative, providing a traditional environment that didn't shake everything up with redefined paradigms and "enhanced user experience" shenanigans. Xfce was the go-to desktop for people who primarily used GTK-based applications but just couldn't get on with Gnome 3.

But then *Mate* arrived, providing a continuation of the Gnome 2.x codebase and its more conservative interface, leaving Xfce as "that other desktop". Yet Xfce has been able to fill an important role: it's halfway between a simple window manager and a fully-fledged desktop environment, providing a consistent set of tools and configuration utilities, without expecting developers to write applications against its libraries (as in KDE and Gnome). Xfce 4.12 has been a long time coming – nearly three years – and promises to fix some of the major quibbles present in previous releases.

All being well, Xfce 4.12 will make it into the next round of distribution releases, including Xubuntu 15.04. If you're desperate, you should be able to get it via a rolling-release distro such as Arch; we installed it from Arch's Testing repository, but it will probably have migrated into the main repositories by the time you read this. If you're feeling especially ambitious and have a bit of time on your hands, you could try building

the whole desktop from source by following the instructions at <http://docs.xfce.org/xfce/building>. It's a good idea to remove any previous installations beforehand if you elect to do this.

New features

So, what's new in this release? From a cosmetic standpoint, one of the biggest improvements is rudimentary HiDPI support. If you're unfamiliar with the term, HiDPI refers to very high resolution displays – such as those used on the MacBook Pro "retina" models, the Chromebook Pixel, and various other recent laptops. The Linux desktop is very much a mixed bag when it comes to HiDPI, with some applications looking fantastic, some looking rubbish, and most falling somewhere in between. Gnome and KDE have a chunk of work on HiDPI, and now Xfce has made a start by including a couple of window decorations that work better on those displays. It doesn't magically make everything better, and you'll still have problems with tiny (or blocky) icons in places, but progress is being made.

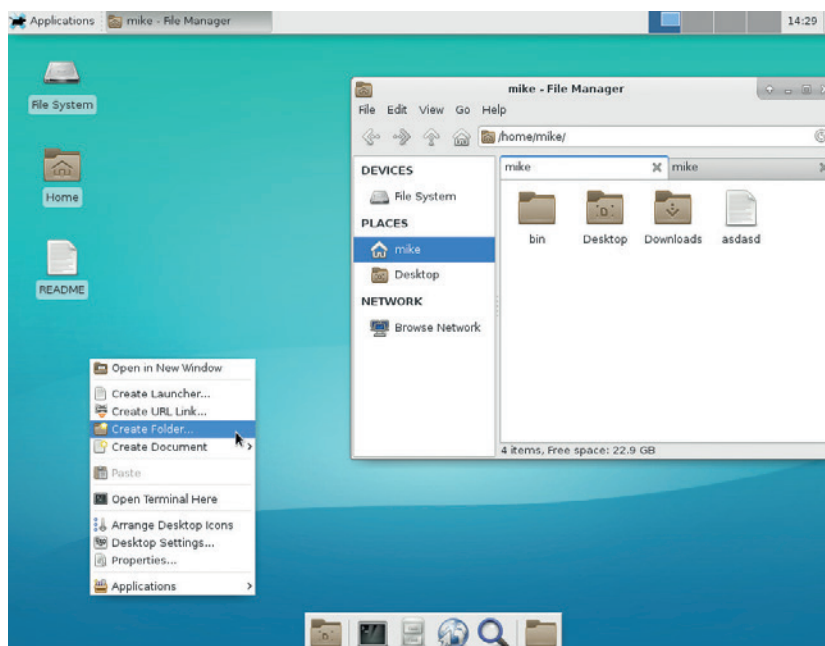
Switchy switchy

Meanwhile, Alt+Tab switching has been greatly improved. If you enable the *Xfwm Compositor* (Menu > Settings > Window Manager Tweaks > Compositor), you'll see that there's an option for 'Show Window Preview In Place Of Icons When Cycling'. This generates thumbnails of windows when switching with Alt+Tab, which is tremendously useful if you have multiple windows open from the same application. If you prefer a vertical list view, that's also now provided under the Cycling tab (see 'Cycle Through Windows In A List').

One notable feature missing from previous Xfce releases was workspace-specific backgrounds. This has been a common request among Xfce users, and helps if you split your workspaces into different categories. For instance, if you have one workspace devoted to work, another to your personal stuff, and a third to something else, it can often help to have different wallpapers for each one, to really distinguish between them. So now, in the Desktop Settings tool, there's an 'Apply To All Workspaces' checkbox – untick that and drag the tool to a different workspace to set its own specific background.

Similarly, if you work with a multi-monitor setup, you'll be chuffed to discover that choosing a layout is now quicker and easier than before. When you plug in an extra display, a box pops up asking which layout you want (eg mirroring the current display, or extending it to the right). You can then fine-tune the

Xfce 4.12 doesn't look radically different to previous releases, but it has new HiDPI window decorations and compositor improvements.



specifics in the usual Display configuration tool. The Appearance tool has also been updated, now showing previews of colour schemes and icon themes.

Another popular request from *Xfce* fans has been intelligent panel hiding. Up until now, there have been two options in the desktop: show panels permanently, or hide almost all of them except for a tiny sliver. When you mouse over that sliver, the panel pops up. *Xfce 4.12* introduces an intelligent panel mode: panels are visible all the time, unless you drag a window over them, in which case they disappear. Move the window away and the panel pops back up. It's only a small touch, but useful if you don't have much screen space to play with.

Power management has seen major improvements, with a redesigned power settings dialog and a new panel plugin which lets you change screen brightness with a single click. (Note: if you're been running a recent Xubuntu release, you may have seen these improvements already, as Xubuntu has included some features from development snapshots of *Xfce*.) If you use the *Light-locker* screensaver program, you can also configure it from inside the power settings dialog. It all feels much more cohesive than in previous releases, and brings *Xfce* up on a par with KDE and Gnome on laptops.

Hover hand

Many of the tools and utilities bundled with *Xfce* have received updates. The *Thunar* file manager, for instance, now supports tabs, so it's possible to have multiple filesystem locations open without having to juggle several windows. During drag-and-drop operations, you can hover the cursor over a tab and its contents will be displayed – a neat little touch.

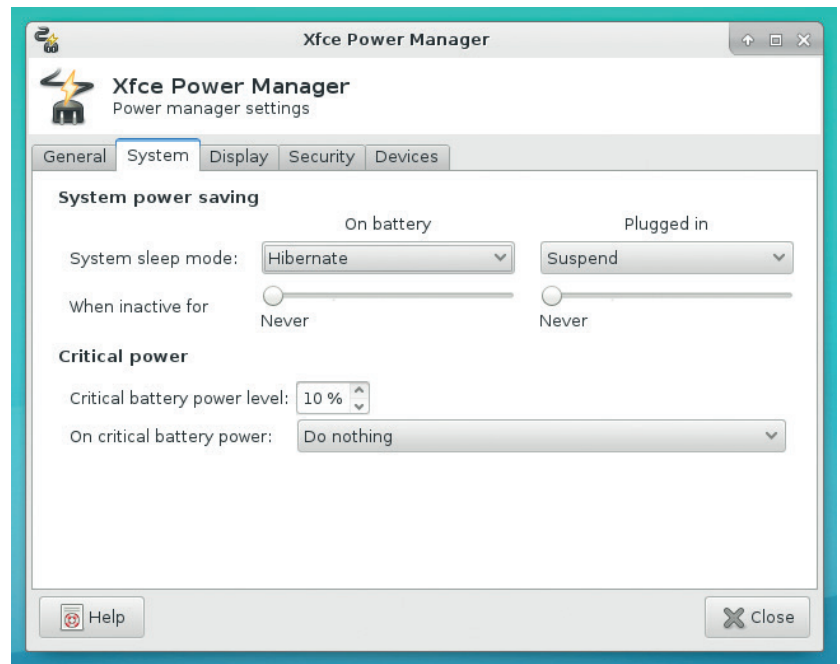
Another useful addition is the ability to select multiple items in *Thunar* and bring up a single properties dialog, showing a total size of all the files and directories.

One of the long-term goals of the *Xfce* team is to port the desktop environment to *GTK 3*. This is a big job, but will bring many benefits such as proper HiDPI support across the whole desktop. The bulk of *Xfce 4.12* is still based on *GTK 2*, but some utilities have made the switch, such as the *Mousepad* text editor and *Parole* media player. The latter program can now support multiple video back-ends and nifty controls that slide over and disappear after a timeout.

Finally, the *Xfce Task Manager* has seen a bunch of improvements, with a better tree-like view of processes, the ability to filter by name, and a port to *GTK 3*. In the coming months, we can expect more of *Xfce's* bundled programs to be spruced up and ported to the newer toolkit, even if it takes longer for the desktop as a whole to make the switch.

So, bearing in mind all these changes, is it worth upgrading if you're happy with *Xfce 4.10*? We'd say

“Is it worth upgrading to Xfce 4.12? Yes – absolutely, unequivocally yes .”



Good news for laptop users: the power configuration has been revamped, separating button and lid actions from overall system behaviour.

yes – absolutely, unequivocally yes. Sometimes we're reluctant to recommend major upgrades when software is very fresh and bugs have yet to be found, but *Xfce 4.12* has been in development for years, and many of its components have already received plenty of testing in recent Xubuntu releases. We found it to be rock-solid, with no noticeable regressions, and wholeheartedly recommend the upgrade. *Xfce* still doesn't have all the bells and whistles of KDE, but for

users who want something lighter on the RAM banks, it provides all you need.

There's also something else important to point out about *Xfce*. On the project's website, the developers

make it clear that most of the screenshots were taken on OpenBSD, highlighting that the desktop is not tied to a particular operating system. As Gnome and KDE make moves to become more dependent on *Systemd*, and thereby potentially limiting themselves to Linux in the long run, *Xfce's* platform independence could be a major winning point for users who want the same interface across Linux, the *BSD flavours, and even other variants of Unix. We're not slamming *Systemd* here – it brings several benefits to the table – but *Xfce* could turn out to be the most portable desktop of all. And that, for many users, is a big deal. 🐧

LINUX VOICE VERDICT

Fast, light, attractive, customisable and portable – the new features in version 4.12 bring *Xfce* close to perfection.



VLC 2.2

Ben Everard has a library of several thousand high-definition cat videos. He just needs a media player that can do them justice.

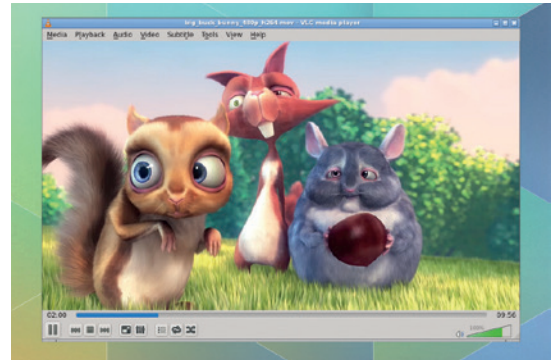
DATA

Web
VideoLAN Organisation
Developer
www.videolan.org
Licence
GPL and LGPL

To play a video at 30FPS on a fairly modest 1080p screen requires changing over 62 million pixels each second. If that sounds a lot for a PC, consider that it can be done on a low-power ARM chip in a smartphone. Playing video smoothly is something that we take for granted, but takes a staggering amount of computing power to perform because the pixels don't just have to be changed: the new values are calculated based on previous values, movements and spatial frequencies. It's only possible to play videos because we have highly optimised software to encode and decode videos (known as codecs), and these are designed to take maximum advantage of special video hardware in the computer.

Before VLC became popular, playing video on a computer usually involved manually finding and installing the appropriate codecs – provided they could run on your hardware. VLC was the first piece of video playing software that 'just-worked' with almost every video standard, and it made watching video on computers easy.

The biggest improvement in this area in the new release is the improved support of two new, high-efficiency codecs: VP9 and HEVC (also known as H.265). Both of these offer significant filesize savings over older codecs. VLC has brought this 'just-works' approach to other platforms. As well as supporting just about every desktop OS currently developed (including some unusual systems such as QNX and Syllable), VLC is rapidly expanding into the mobile and



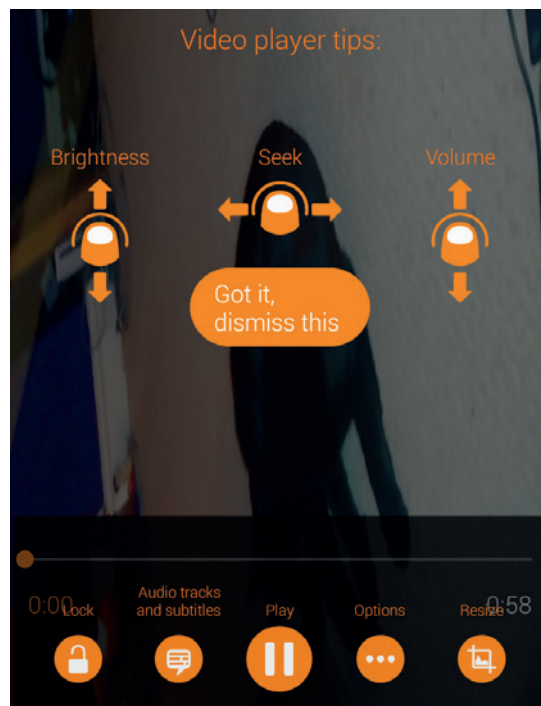
VLC is the most downloaded piece of software from SourceForge, with almost 900 million downloads.

embedded world. There are now releases for Android (and Android TV), Windows Phone, WinRT and iOS. Not all the mobile versions are yet considered stable, but this release brought the first stable version of *VLC For Android*, which comes with a redesigned interface that should fit in better with other Android apps. If you're installing on Android, make sure you get the *VLC For Android* (without the word Beta in the title) app by Videolabs, as there are a few others that use the same traffic cone icon.

Improved performance

For low-power devices such as the Raspberry Pi, this new version brings improved OpenMAX (Open Media Acceleration – a standard for hardware acceleration) support, as well as improved efficiencies in the acceleration generally, which should improve performance by reducing load on the CPU during playback. However, for all the advances behind the scenes, the interface for VLC remains functional rather than stylish, at least on the desktop. You can install skins to make the application look different (you can get these from www.videolan.org/vlc/skins.php), but none of these fundamentally change the method of interacting with the application.

Version 2.2 is another solid release for VLC. It fixes over 1,000 bugs and brings support for more formats and more devices. Most of the updates are behind the scenes, so it's well worth updating even though you may not notice the differences straight away. If you're an Android user, this release is an even more important upgrade. 📺



The touch controls to VLC on Android can be a little confusing, but it provides an instructions overlay when you start a video.

LINUX VOICE VERDICT

VLC remains the most capable media player available on any desktop OS.



Krita 2.9

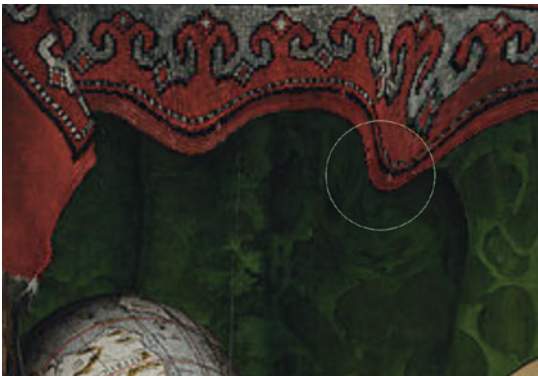
Graham Morrison looks at the biggest Krita release ever. And refrains from drawing Richard Stallman!

If you were in any doubt that crowdfunding can work (and you shouldn't be if you're reading this magazine) the team behind *Krita* has just pushed out a huge update – thanks to crowdfunding. Most of the additional features in this update are thanks to a hugely successful Kickstarter campaign that concluded in July 2014. The team raised €19,995 to fund two developers, Dmitry Kazakov and Sven Langkamp, to work on a set of specific features. As a result, eight months' work has yielded the biggest update *Krita* has seen.

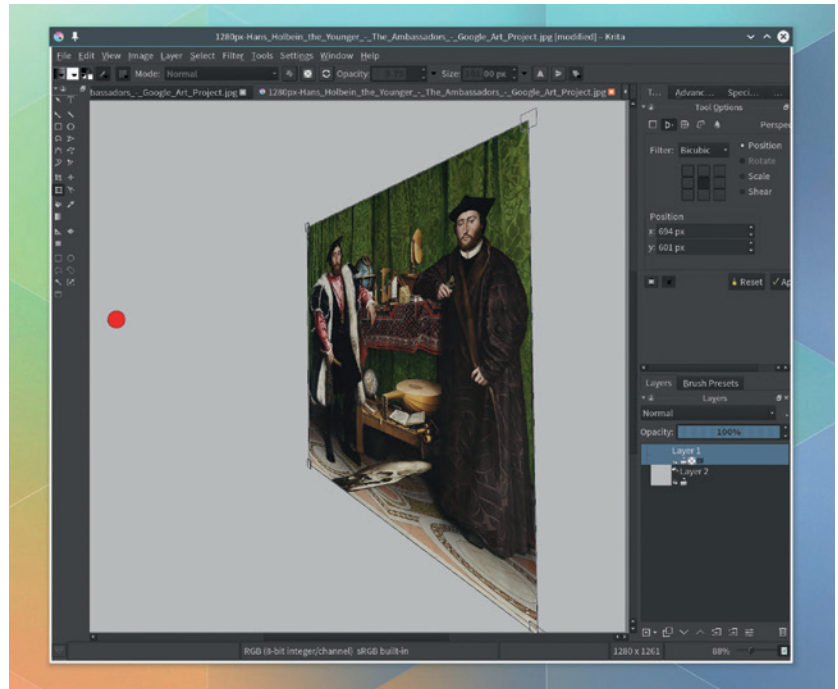
At the top of our list of favourite new features is the incredible perspective transformation tool. This enables you to warp, shape and manipulate the pixels within a selection by dragging the corners of the selection as if they were located in 3D space. The transformation will even show the vanishing points for edges, and it's a brilliant tool for fixing or matching selections that need to be pasted onto something that isn't flat. It's also great fun – just like playing with a similar feature in the old *Deluxe Paint*, except that the final output in *Krita* looks perfect. Similarly, with the cage transformation, you can create anchor points and move these around to adjust the selection they contain. These transformations go particularly well with the new options for the perspective assistant. These snap your drawing cursor to a specific plane within 3D space, allowing you to sketch perfect cubes, or roads, or 1970s-style string art.

We are the children of Tony Hart


Also falling into the fun and useful category is the liquify tool. This allows you to drag sections of an image as if they were stretched across a rubber sheet. Slow dragging will leave subtle distortion along the path of your stroke – very useful for adding waves to curtains or cloth, or even reducing them, as well as lots of other creative distortion. All of these



The transform effects render such high-quality changes to the image, subtle use makes their use difficult to see.



transformations are previewed in real time thanks to the new transform masks, and we had no problem using them, even though finding the control to activate them was a little unintuitive. Not as exciting, but just as useful, you can now work on more than one project within the same window, as new projects are added to a separate tab. You can switch this view to a preview of whatever images are loaded, which is useful if you're working on a sequence.

There are many, many other features added to this release; RAW support, improved *Photoshop* compatibility, exposure controls and new filters. We also found this release to be significantly less CPU intensive. We did experience a few crashes, but that was probably because we were pushing the 3D transformations too far. As you can probably tell by the kind of examples we've been using, *Krita* is going beyond being an exceptional drawing tool and entering the realm of image editing and processing. We find ourselves launching *Gimp* far less – especially as *Krita* has always had CMYK support, for example, and its modern, adaptable user interface leaves *Gimp* way behind. We think it's brilliant. 

You can adjust selected areas in 3D space, and use assistants to help you draw onto the projected canvas.

DATA

Web
<http://krita.org>
Developer
Team Krita
Licence
GPLv2

LINUX VOICE VERDICT

This is a huge release. Perfect for amateurs and even professionals working with any kind of imagery.



Inkscape 0.91

For a major new release, this leading vector graphics editor really should be at version 1.0 or 2.0, thinks **Graham Morrison**.

DATA

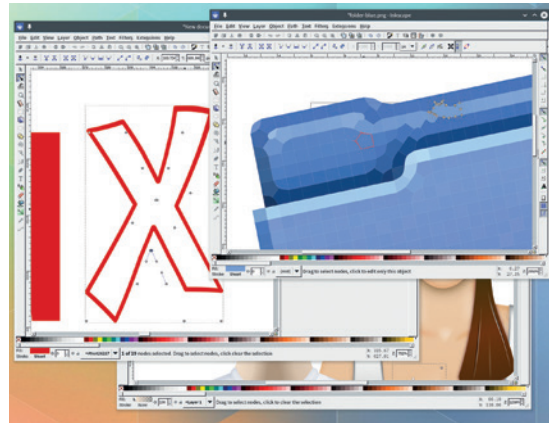
Web
inkscape.org
Developer
The Inkscape Team
Licence
GPL

Version 0.91. Surely a minor update? Not worth a review? That's what most users would think, were they not primed to think like open source developers. Of course, *Inkscape 0.9* is a major update and the first for four years. And that's not even the half of it. Literally. This version number scheme isn't a vestige of some old system, because the previous major release was 0.48 from 2010. The development team has since jumped half a unit to reflect "its maturity as a vectors graphics editor," rather than as a tool for humbly editing SVG images, which was *Inkscape's* original intent. So 0.91 it is. The first major release of Inkscape for four years.

Putting version numbers aside. *Inkscape* is now the *de facto* open source vector graphics editing application. That's a huge achievement, and it's a genuine alternative to Adobe's costly Illustrator. We use *Inkscape* for creating and editing our own scaleable graphics. It's very powerful and without any doubt, easily capable of professional output.

But you need to know what you're doing to get the best out of it. The main reason for the long delay between releases is that the rendering engine has been completely replaced with the *Cairo* vector graphics library. This is the same engine used by Gnome 3.x and numerous other projects, and *Cairo* is now mature enough to handle much of the functionality required by *Inkscape*. It's faster and more efficient, although we still had problems with large and complex illustrations, or with certain filters that added greatly to the number of points.

There are so many additional features that we don't quite know where to start, so we'll focus on those few that make most sense to us. Firstly, you can now move layers around, just as you've been able for a couple of decades in *Photoshop*. You can also drag and drop layers on top of one another to create easily navigable groups. This feature alone is worth




Vector Lovers is both an electronic music ensemble and a collective term for everyone who thinks *Inkscape* is ace.

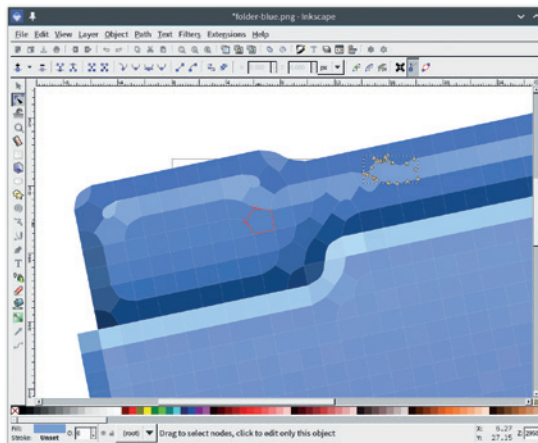
the upgrade, because we typically use many different layers for our own projects.

The new measurement tool is also awesome. Click and drag away from a point and the cursor turns into a protractor, complete with the distance and angle of the new point against the old. This is brilliant for design, because you often want to hit specific points.

Walk like an Egyptian

Another function we use is 'Trace Pixel Art'. This will turn a low-resolution bitmap image, such as a sprite from an old video game or an icon, into a matrix of Bézier curved scaleable cells. It's not practical in the way that the old Trace Bitmap function attempts to be (fortunately, this tool is still there), and definitely can't be used for large images, but it does create a lovely pixelated version of the original that looks great in print. We also like the the new PowerStroke. This is one of the path effects that traces the route across your nodes and enables you to adjust the thickness of that path around the nodes, and the new version allows you to click on special nodes to adjust this thickness in a far more intuitive and creative way.

Vector images have never been so important. High-DPI screens and hardware acceleration for rendering are turning SVGs into the new native image format, especially for GUIs. And there's nothing close to *Inkscape* – it's a brilliant piece of software that's just taken a huge step forward. 



The new Trace Pixel Art feature in action. Converting a KDE pixel icon into bubbly vectors.

LINUX VOICE VERDICT

Just think how rubbish your desktop icons would look if we didn't have *Inkscape*.



IPython 3.0

Interacting with Python may sound like a recipe for asphyxiation, but with the right tools, it needn't be, finds **Ben Everard**.

IPython is a project to improve the interactive shell for the Python programming language. That means it's not designed for writing programs that run as a whole, but for executing code line by line as it's entered.

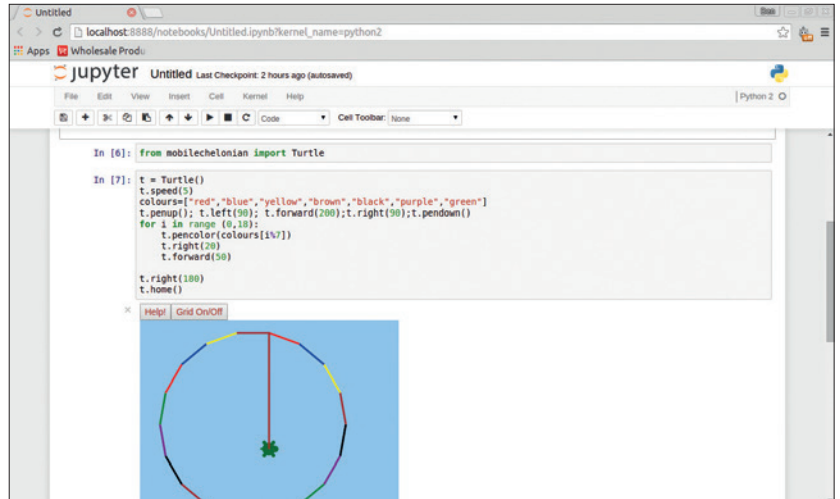
IPython can be run in a terminal (by starting it with **ipython console**), in its own window (by starting with **ipython qtconsole**), or in a web browser (by starting with **ipython notebook**). Functionally, all of these work the same; they just use a different user interface.

The main focus of IPython isn't on writing software, but in using the interactive shell to explore code and data. IPython really shines in areas like data analysis where the user wants to use the power of the Python tools (such as *SciPy* and *Matplotlib*) to interactively analyse a data set. Compared with the standard Python shell, IPython offers far more functionality for tracking what you're doing, introspection and displaying media. IPython also makes it easier to parallelise your processing in a standard manner, which helps speed up the analysis of large data sets.

Serious work

Exporting sessions is another area in which IPython shines, so you can share your work with colleagues. This can be used for collaboration, but it's also used in the academic community for presenting results, and some books have even been published in the IPython format (for example *Probabilistic Programming and Bayesian Methods for Hackers*: <http://bit.ly/1ta2E3y>).

The project is in the process of changing. The features that made IPython so useful for working with Python are also useful for other interpreted languages, and the front-end has now split in a language-agnostic way. You can already use other kernels (ie languages) beyond Python in IPython, and version 3 will be the last release to be called IPython. In the future, it'll be called Jupyter – a contraction of Julia, Python and R, the




three key scripting languages that the software will target. IPython will live on as the back end, and you'll still be able to use Jupyter in much the same way as IPython works today. The two additional languages are both targeted at a scientific and analytic audience.

The browser-based notebook is already branded Jupyter, and there are some new poly-lingual features. For example, you can create a new notebook using a different kernel from within the web app itself where previously you had to start a new notebook server.

As well as using the language of the kernel, IPython includes some features for interacting with the host OS. Any line starting with a % sign is processed by IPython itself. This creates a mini-language of magic (in IPython terminology) commands for doing things like interacting with the filesystem and the host OS that can be run alongside the kernel language.

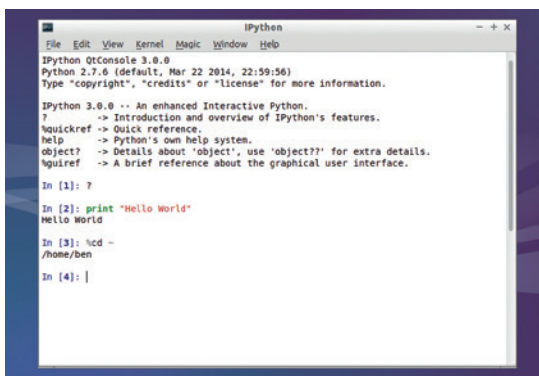
Version 3 is mostly backwards compatible with earlier versions, but there are a few changes that could cause problems. The ones you're most likely to hit are in widgets for the browser-based notebooks. You can take a look at the migration guide at http://ipython.org/ipython-doc/3/whatsnew/version3_widget_migration.html.

IPython had previously established itself as the best open source numerical and data analytics platform. Version 3.0 cements this lead, and the growing acceptance of new languages in the project make it accessible to a larger audience. 

You can try IPython without installing it by going to try.jupyter.org.

DATA

Web
www.ipython.org
Developer
IPython Development Team
Licence
BSD Licence



The Qt interface now has improved support for Qt 5, making it a better fit for more modern interfaces.

LINUX VOICE VERDICT

IPython continues to set the standard for interactive programming environments.



The Secret Life of Bletchley Park

Ben Everard gets a timely reminder to use secure passwords.

The cryptography and computer science of Bletchley Park are well known, at least now. However, how did this code-breaking institution come about, how did it run, and how did it remain secret for so long? In *The Secret Life Of Bletchley Park*, Sinclair McKay talks to the people who were there about what went on.

The staff of Bletchley – many of whom were plucked straight from university – stayed with local families, as there was only enough space on the main campus for the offices. Thrown together in the chaos at the start of the war and left to get on with the business of cracking uncrackable codes, the staff of Bletchley Park developed a unique culture. McKay takes the reader into this environment and follows the park through its triumphs and setbacks during the war.

The Secret Life of Bletchley Park charts the changes at the park chronologically, but

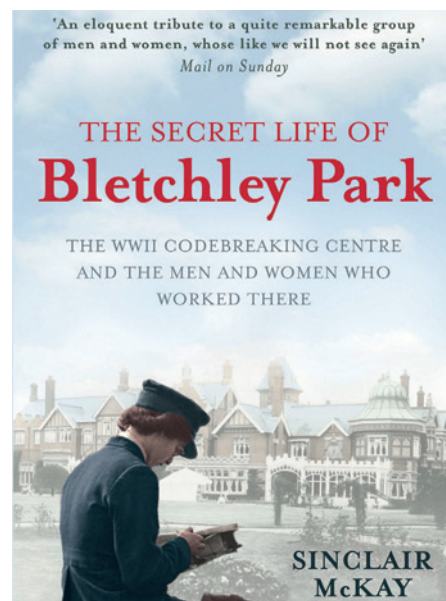
each period is used to investigate a particular aspect of the Park. Hiring staff, accommodation, entertainment and food each came with their own challenges in rationed, war-time Britain.

The Secret Life of Bletchley Park will fascinate anyone who wants to know more about the behind-the-scenes running of the famous institution, though people looking for in-depth technical details of the work at Bletchley would be better served by a different book.

LINUX VOICE VERDICT

Author Sinclair McKay
 Publisher Aurum Press Ltd
 ISBN 978-1845136338
 Price £8.99

The Secret Life of Bletchley Park follows the human side of the early cracking pioneers.



Bletchley Park is now a museum marking the achievements during the second world war.

Jagged Alliance 2

Ben Everard dons his rose-tinted glasses and gazes back at games past.

Jagged Alliance 2, or *JA2* to its fans, is a turn-based role playing game released in 1999. While it isn't generally considered a commercial success, it did develop a loyal fanbase, and in 2001 it became the first game that Tribsoft ported to Linux. In this book, Darius Kazemi looks back at the game, how it was developed, and how the various components came together to make the finished game.

Jagged Alliance 2 (the book) follows development of the game chronologically from the first game in the series to the various mods and expansions. Through this canvas, it explores how games were developed in the 90s, the business of games (as it was then), and the commercial pressures on games companies.

This all happened during the period when the games industry was transforming from the 80s, when a talented individual in their bedroom could create a commercially successful game, to the industry of the 2000s, when creating a game required millions of pounds and an extensive team.

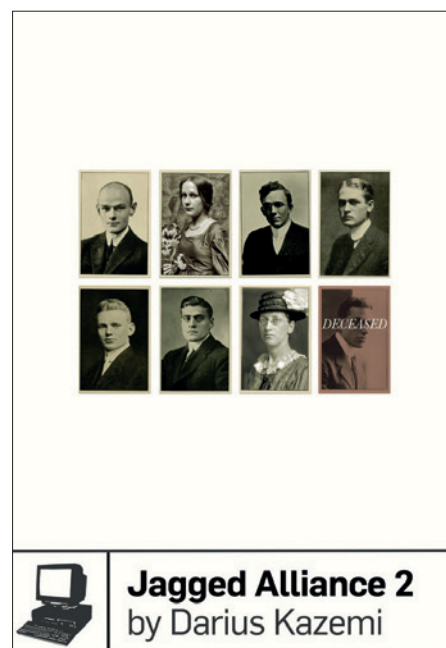
The modern Indie movement has reversed this trend.

Kazemi talks to a lot of the people behind the game from developers to designers to voice actors to publishers, and together creates a complete picture of what went into it. Games development has changed beyond recognition in the intervening years both from a technical and business perspective, so this can't really be viewed as a particularly instructive book for people looking to get into the industry. Instead, it's a little slice of history that's been saved for those of us that fondly remember an earlier era in the history of PC gaming.

LINUX VOICE VERDICT

Author Darius Kazemi
 Publisher Boss Fight Books
 ISBN None
 Price \$4.95 (ebook) \$14.95 (paperback)

Games are now part of our culture, and can't be forgotten when technology moves on.



Jagged Alliance 2 is part of Boss Fight Books, a crowdfunding series based on classic computer games: <http://bossfightbooks.com>.

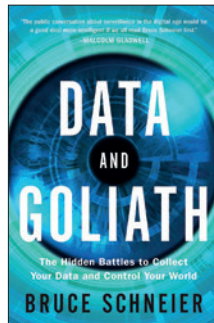
Data and Goliath

Mike Saunders dives into the debate on mass surveillance.

When Bruce Schneier talks, people listen. He's one of the most prominent experts on security and cryptography, and his latest book aims to bring the debate about mass surveillance – from both governments and companies – to a wider audience.

Data and Goliath points out that mass surveillance and infiltration isn't just a problem for our personal rights, but it's also severely impacting the ability to do business in the modern age. He notes several companies that have recorded major losses in profits since the Snowden revelations – after all, who wants to do business with a country or company that sneaks backdoors into everything?

What makes *Data and Goliath* a good read is Schneier's understanding that there's inevitably a trade off in this world. He accepts that there are legitimate needs for government surveillance, and also that many business models are based on having access to customer data. So he



Of the book's 370 pages, 120 are notes on the main text.

puts forward a series of solutions to the problem – practical and sensible – which could help us to find a middle ground. Will anyone listen though?

LINUX VOICE VERDICT

Author Bruce Schneier
 Publisher Norton
 ISBN 978-0-393-24481-6
 Price £18.45/\$27.95

Nothing shocking for geeks, but a book that all politicians and CEOs should read.



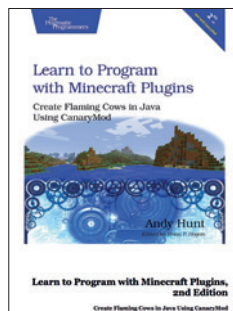
Learn to Program with Minecraft Plugins

Graham Morrison pretends he's working when he's really playing.

Any book that starts with a reference to *Colossal Cave Adventure*, a game from 1976 that was purely text-based, is already winning. *Learn to Program With Minecraft Plugins* presents the challenges and creativity of writing your own *Minecraft* plugins in the same way you might play *Colossal Cave*, starting with a brief overview of both the command line and text editors before diving into some code to build a house.

We love the way in which programming theory is interspersed within the process of doing cool stuff with *Minecraft*, and the layout and logical flow of the writing is excellent. If you're into *Minecraft*, it's a brilliant way of harnessing your imagination to learn programming.

However, there's one huge caveat: make sure you get the second edition. This is because the first edition relied on the *Bukkit* library, which has since disappeared. The second edition replaces this with *CanaryMod* and does the same thing.



Make sure you get the Second Edition!

Which means the only slightly negative aspect in our view is our confidence in its future, now that Microsoft owns the IP to *Minecraft*.

LINUX VOICE VERDICT

Author Andy Hunt
 Publisher Pragmatic Bookshelf
 ISBN 978-1-941222-94-2
 Price £19.50

A great way of combining your love for *Minecraft* with a desire to learn to code.



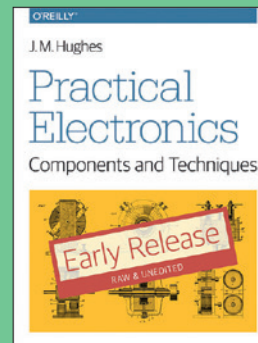
ALSO RELEASED...



If you were wondering what to get Ben for his birthday...

Blockchain

Cryptocurrencies are here to stay. We're all going to need to understand how blockchains work and how they can guarantee the validity of a transaction if we want to safeguard our finances. Even though it's a complex subject, it's easier to understand than banking.



Perfect for software developers who want to build stuff.

Practical Electronics (Early Release)

We wish we'd been taught some electronics at school, rather than Home Economics. Fortunately, this title looks perfect for filling in the gaps in our knowledge. It's raw and unedited in its current form – a book in beta, but it means you get hold of the content now.



Kathy Sierra has written another book. That is all.

Badass: Making Users Awesome

We've been looking forward to this for a long time. It's the return of Kathy Sierra and her wonderful insights into what makes things the way they are. Here she answers one simple and compelling question: why does one similar and competing product outsell the other?

GROUP TEST

A compulsive data hoarder, **Mayank Sharma** is constantly on the lookout for more spaces to fill.

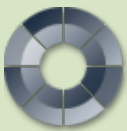
On Test

FreeNAS



URL www.freenas.org
 VERSION 9.3
 LICENCE BSD Licence
How does the most popular NAS distro stand up to the competition?

NAS4Free



URL www.nas4free.org
 VERSION 9.3
 LICENCE BSD Licence
Can the spiritual continuation of the original FreeNAS code, top the real thing?

Open Media Vault



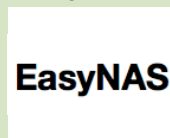
URL www.openmediavault.org
 VERSION 1.9
 LICENCE GNU GPL v3
Led by a former FreeNAS developer, does this Debian-based distro have what it takes?

Openfiler Community Edition



URL www.openfiler.com
 VERSION 2.99
 LICENCE GNU GPL v2
Will this open source version of a proprietary solution pound the rest?

EasyNAS



URL www.easynas.org
 VERSION 0.5.3
 LICENCE Several free software licences
Just a clever name?

Turnkey Linux File Server



URL www.turnkeylinux.org/fileserver
 VERSION 13.0
 LICENCE GNU GPL
What's a file server doing rubbing shoulders with NAS distros?

NAS distros

What's a terabyte to a data connoisseur? If you're like us, you probably have more data than spare USB ports. While external drives are a great way to quickly and conveniently add extra storage, they have their drawbacks. For one, their data retrieval capabilities are restricted to the computer they are connected to. This might work for individual users with single PCs but isn't a practical solution for a household with a variety of devices.

To add flexibility to your data storage and retrieval you need to use a network-attached storage (NAS) solution. With a NAS you can essentially share the storage with everyone on the network. While you can pick a prefabricated NAS box from PC World, it doesn't take much effort to build one yourself. In this feature we'll test some of the best NAS solutions that offer you the features and flexibility of a commercial NAS minus the cost of proprietary software.

Almost all NAS solutions also offer additional advantages. Instead of simply pooling together the attached disks, they let you arrange the available space into different RAID configurations and give you control over how you want to store your data. You can decide to spread your data across the drives or create different levels of redundancy for an effective backup solution. Most NAS solutions support a variety of protocols and can be accessed from multiple operating systems and devices. Some even allow access to remote machines outside the network.

Many NAS solutions can do a lot more than just back up and restore files – you can extend them with plugins to do a variety of tasks. Some enable you to stream media to computers and others devices. Others can hook up with apps and services and allow them to use the NAS for storing and retrieving data. Read on to find out which NAS solution works best for you.

“With a NAS solution you can share the storage with anyone on your network.”

Evaluating network attached storage

The most popular NAS solutions aren't based on Linux, but rather on FreeBSD, which isn't necessarily difficult to install but has its own peculiarities. We'll also pay special attention to their respective administration avenues. Virtually all of them have graphical web-based admin interface and we'll rate them for their ease of use and flexibility.

Although the NAS solutions on test support a variety of features, since they're primarily handling data, we'll keep an eye out for associated capabilities such as data encryption and redundancy. Solutions will also be awarded for the number of useful plugins and extensions and their support infrastructure.

Build a NAS box

Let's go shopping.

A NAS solution requires both software and hardware. While this feature will help sort out the software bit, you'll need to put together the hardware for your DIY NAS box. The most important part of the NAS box is the storage. Although you can use a solitary hard disk, you should definitely start with at least a couple. You'll also need a smaller one on which to install the NAS software: most NASes requires a

disk exclusively for themselves, which makes it pointless to use a 1TB disk for installing the NAS distro. Some NAS distros can also live off USB disks.

You'll also need a mini-ITX motherboard to power the NAS box. Look for one that supports multiple SATA drives. Memory is important too, and since it isn't as expensive as it used to be, we recommend you pick up at least 8GB to be future proof.

If your requirements are modest, you can even turn a Raspberry Pi into a cheap NAS server. Just attach a large portable USB disk to the RPi and install and configure Samba on it to make it accessible from anywhere on the network. If you need redundancy, you can attach another USB disk and duplicate the data in one to the other with a simple **rsync** command and then make the process automatic with **cron**.

OpenMediaVault

Lock and load.

The OpenMediaVault (OMV) NAS distro is designed for small businesses and home users, and has modest hardware requirements. Installation is pretty straight-forward since OMV takes over the entire disk. This might seem like an odd choice, but you can install OMV on to a removable USB disk as well. Due to its specialised nature, OMV lacks the baggage of a normal distro and can easily fit inside a 4GB USB disk.

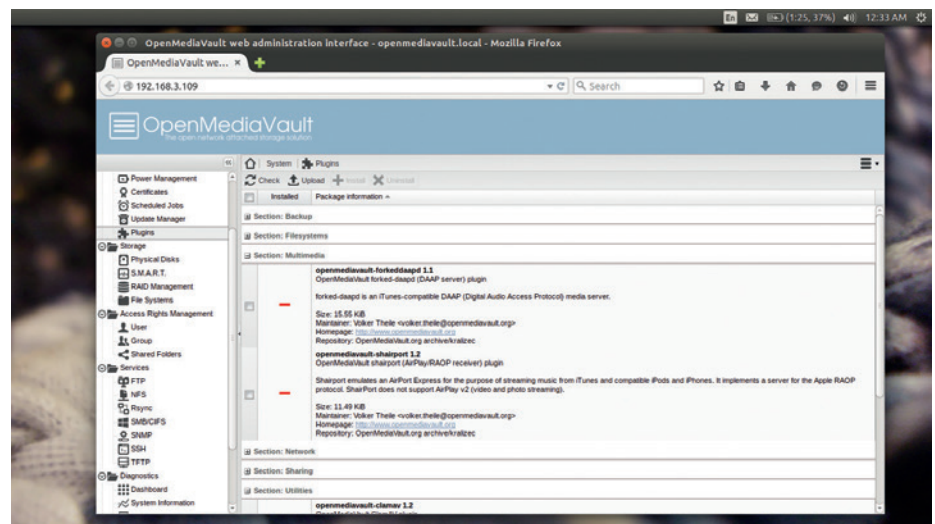
Once it's up and running, you can manage the distro from its browser-based administration interface, which is well laid out, with the options listed in a logical manner. OMV will detect all attached disks and even lets you wipe them securely. You can also enable SMART monitoring for the disks and schedule tests.

You can use the disks attached to the OMV NAS individually or assemble them in a RAID array. OMV defaults to RAID level 5 but supports all RAID levels. You can also format the individual disks or the RAID device from the web interface.

OMV can create and manage EXT3/4, JFS, and XFS filesystems. You also get the option to assign disk quotas to individual users, and the distro has ample options for managing users. There's also the option to import multiple users in a particular format, and you can define per-user access permissions for every shared folder.

All-rounder

Once the storage has been added, you can access the NAS from anywhere on



OMV, built on Debian Wheezy, is chock-full of features and can easily take on new ones with plugins.

the network using a variety of ways. OMV supports various popular protocols and services, including NFS, SMB/CIFS, FTP, TFTP, SSH, *rsync* and more. Each service has its own configuration and management screen. You can configure various aspects of each service before enabling them, and can define the shares for the different services individually.

You can conduct regular system maintenance tasks such as installing updates from the web interface. The distro has custom command-line scripts for tasks such as upgrading to new releases, and you can schedule them via the web interface.

One of OMV's strongest suits is its ability to take on new features with plugins. The distro ships with 11 officially supported plugins and you can add a variety of third-party plugins hosted on **omv-extras.org**. The officially supported plugins add a couple of features that ship as standard on

some NAS distros but are missing from the base OMV installation. For example, OMV doesn't let you pool multiple disks into a logical volume by default. Similarly, OMV can't interface with a directory server, but with the LDAP plugin it can be made to fetch user authentication information via LDAP. Then there's a plugin that lets you stream the music stored on the NAS and another that can automatically synchronise a shared folder to a plugged-in device.

The project has plenty of support infrastructure, with enough documentation on its wiki and an active forum board. You can also sample OMV's admin interface using the demo installation on its website.

VERDICT

A feature-rich NAS distro that's easy to deploy and manage.

★★★★★

Turnkey Linux File Server

Go go gadget.

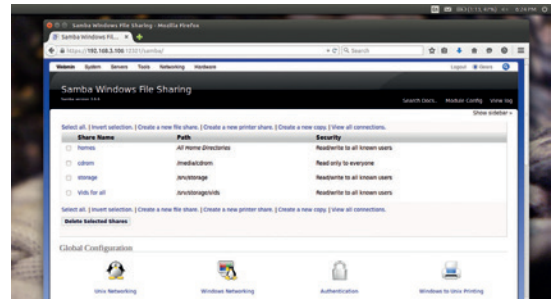
Turnkey Linux produces a range of self-contained distros all based on the latest stable Debian Wheezy release. You can download all Turnkey Linux distros as installable live ISO images or as virtual disks optimised for various virtualisation platforms such as *VirtualBox*, *OpenVZ* and *Docker*.

The File Server appliance includes a pre-configured Samba installation. Think of it as a bare Debian installation with a fully configured and working instance of the Samba server. The server will show up on your network as soon as you're done installing it. By default, the Samba installation has configured shares for every user's home directory and a public storage area readable and writeable by all users.

The distro also ships with Webmin for managing various aspects of the underlying distro from the browser. The customised Webmin installation ships

with the Samba module with which you can graphically configure Samba. Using the Samba module you can change the default workgroup and Netbios name of the Samba installation as well as add and remove Samba shares and fine tune their permissions. You can also use Webmin to add users to the base Debian distro.

While the Turnkey Linux File Server distro is the simplest to deploy and use, it's also very bare in terms of features. Unlike most NAS distros, it'll let you use the free space on the disk it's installed on. You can also add additional disks and share them via Samba. But to use them together as a virtual volume, you'll need to be familiar with logical



The distro includes *AjaXplorer* for accessing your files from the browser and mobile devices.

volume management (LVM) on Debian. Furthermore, the distro doesn't include the ability to configure RAID like most other NAS solutions on test, nor do you get a multitude of protocols, so you're restricted to using SMB.

“Think of it as a bare Debian installation with a fully configured Samba server.”

VERDICT
A no-fuss distro that'll set up a fully functional file sharing server in no time.
★ ★ ★ ★ ★

Openfiler Community Edition

Open for business.

Openfiler is one of the most comprehensive solutions on test. It's based on the now defunct rPath Linux and is distributed as an installable image for 64-bit machines and also as pre-installed disk images for various virtual machine monitors including *Qemu* and *Xen*.

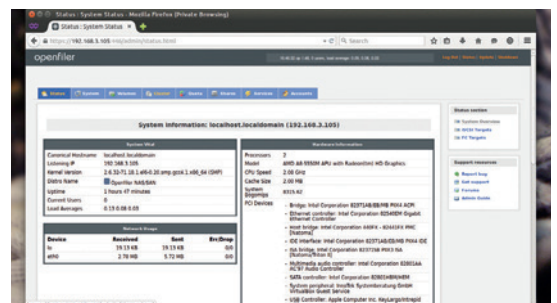
In addition to common NAS features, Openfiler supports a variety of enterprise-specific features such as support for LDAP, Active Directory and authentication protocols such as Kerberos 5. Furthermore, its share management also leaves little to be desired. Besides arranging attached disks into RAID's, it can create an iSCSI target and initiator. One handy feature is the ability to bond multiple Ethernet cards into one network interface for faster data transfers between the NAS server and the users on the network.

However, all these features come at the cost of usability. The Openfiler web

interface is one of the most complex and unintuitive. It's the KDE Control Centre of NAS interfaces. It presents an endless sea of options and sub-options that depend on each other but aren't coherently presented. For example, to pool multiple disks into a simple virtual volume, you'll first have to partition the disks, then hunt for the option to arrange them in a volume group and then find options to create a volume inside them. If you still can't access the disks, that's probably because you haven't enabled and configured the sharing services. The process to arrange the disks in a RAID array is similarly cumbersome.

Where is the book?

To top it all, Openfiler has virtually no freely official documentation, besides an installation guide and a skeletal FAQ with just two questions. If you need an administration guide you'll have to shell



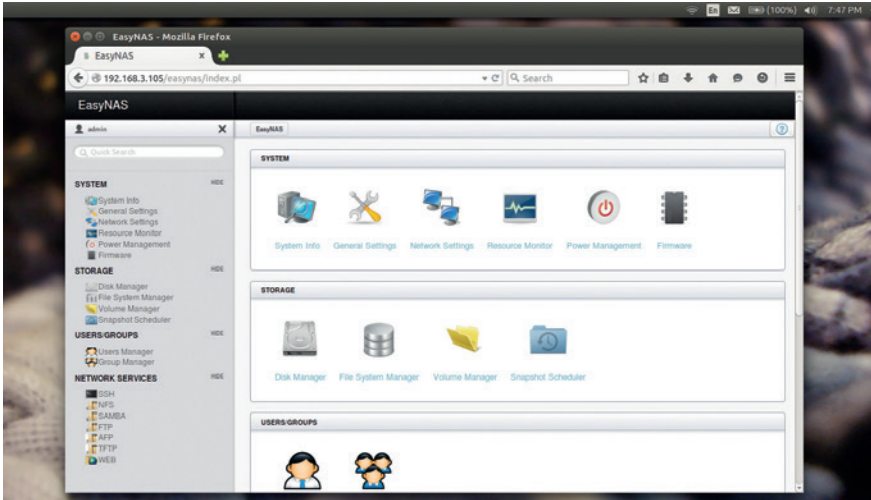
Openfiler switched to CentOS in 2013 and there's a CentOS-based version for testing, but no final release.

out €9.99 (about £7.50). There are several support packages on sale as well, but the community forum board listed on the website is replete with unanswered hails from users.

VERDICT
There is a target segment for Openfiler, but we can't spot it.
★ ★ ★ ★ ★

EasyNAS

Easy does it.



Minor releases can be upgraded to from within the interfaces but to upgrade to a new major release you'll have to reinstall the distro and then mount the existing volumes manually.

Rolling and managing your own NAS server doesn't have to be an involved process. The EasyNAS distro takes away the complexities by making several assumptions on the user's behalf and in essence simplifies the entire process. The distro is built using the online SUSE Studio tool and is based on OpenSUSE 13.2

There's not much to installing the distro. EasyNAS is designed to take over the entire disk, and all you have to do during installation is to point it to the hard disk you want it to take over. As with most NAS distros, you can't use the installation drive to store additional data.

You can carry out some common administration tasks, such as changing the admin password from the console of the installed distro. For setting up the NAS you can use the distro's web administration console. Unlike most other distros, EasyNAS's web interface has many fewer options and is easy to navigate. You can use it to arrange disks in multiple types of RAID arrays and even concatenate multiple disks into one virtual volume.

When creating a filesystem on the disks, you can also choose a compression level. The distro gives you two options (better and faster) without going into details about them. Unlike other NAS distros that support multiple filesystems, EasyNAS only supports the Btrfs filesystem. Also, you can define multiple volumes but don't

get the option to specify their size. In essence, every volume can grow until it takes over the complete disk.

Easy but not kiddie

The distro also has all the essential user management abilities, and while adding users, you can mark them as EasyNAS admins. Furthermore, when you're creating volumes, you can assign the user and group that owns the volume as well as access permissions for them. By default new volumes are automatically added as Samba and NFS shares, and you can optionally add them as TFTP or AFP shares as well (AFP is the Apple Filing Protocol, which is used for sharing files with Mac OS X).

While the distro has some useful features, such as the ability to schedule automated backups of added volumes, it lacks advanced features that you get with other solutions, such as the ability to hook up with a directory server. Also, while the distro supports a variety of protocols and services, you get no options whatsoever to configure them. There is also a web service option that runs a simple web server and enables you to look at and download files from a web browser.

VERDICT

A simple NAS distro that balances the availability of features with reasonable assumptions.

★★★★★

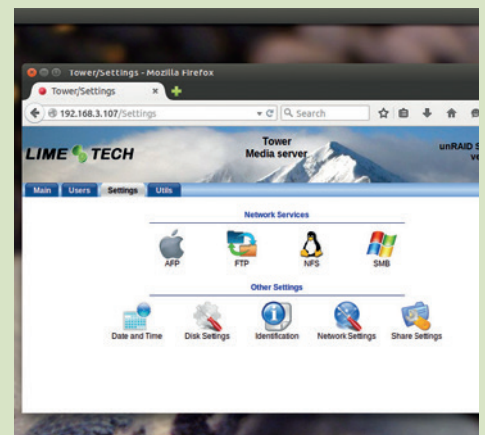
Other options

More NAS solutions abound!

If we're going to recommend a piece of software that you can install at home or at work, we want you to be able to get security and feature updates for it, and for that reason we've featured some of the most popular and actively developed NAS solutions. One potentially useful option that isn't in active development is CryptoNAS. The USP of this distro is that it pays special attention to encrypting the data and ships with multiple encryption algorithms. The Debian-based CryptoNAS is available as an installable distro and also as a Deb package. If you aren't averse to freeware, there's also the Slackware-based unRAID Server. It's available as a USB image and can support up to three disks.

Besides these open source solutions, there are several commercial ones as well. Most charge for their enterprise-specific features, while some charge for support and other conveniences. Server Elements has three NAS products that run entirely from RAM. NASLite-2 is a general purpose NAS solution while NASLite-M2 specialises in streaming media. The company also sells a cheaper NAS solution for home users called NanoNAS.

The open source edition of Openfiler also has a commercial edition that adds a host of enterprise-specific features such as block-level replication and High Availability and support for iSCSI and Fibre channel. There's also Open-E DSS, which is replete with Enterprise-specific features and also has a feature-curtailed Lite version that's available as a free download.



Tower Media Server is based on Linux but tries its best to camouflage the fact so as to not scare away users.

FreeNAS vs NAS4Free

NAS at its best.

FreeNAS is probably the most recognisable NAS distro and one of the elite group of open source software projects that has made a name for itself in the enterprise space. In 2011 its development was taken over by iXsystems, which also sponsors the PC-BSD desktop distro. The new sponsors made changes that didn't go down well with a section of the developers, who forked the project and created the NAS4Free distro.

Besides a common foundation, there are several similarities between the two distros. Both bring the advantages of the ZFS filesystem to the network with the use of popular protocols including SMB, NFS, FTP, AFP, iSCSI and more. Both distros have a similar installation process and can happily reside on a removable USB disk as well.

While FreeNAS is designed to take over the entire installation disk, NAS4Free offers a bunch of options. You can either let NAS4Free completely take over the drive or install it and create a couple of partitions using the remaining space. The second option helps you use the excess space on the installation drive for housing files.

Once installed, the two distros boot to a console with similar administrative options to configure network and change passwords. However, as with other NAS distros the actual NAS configuration is done via a web interface. While the Django-based FreeNAS interface looks modern,

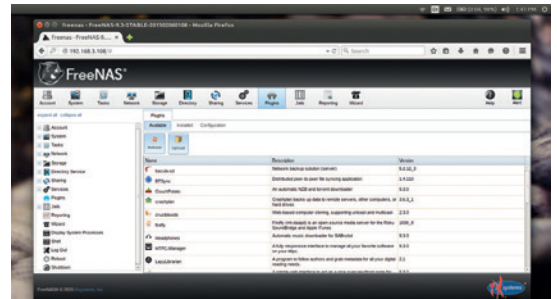
NAS4Free uses a modified version of *Monowall's* aged web interface.

Adding disks is fairly straightforward with both distros. Remember, however, that these distros are both based on FreeBSD, and to use them effectively you need to be familiar with the ZFS filesystem and associated terminology such as zpools and datasets. In this regard, FreeNAS scores over NAS4Free, as its process is more intuitive and visually pleasing and selects good defaults. That said, NAS4Free aids first-time users by pointing to any missed steps. So for example, if you went straight to add a ZFS disk, NAS4Free will tell you that you first have to add a virtual device and point you towards the relevant section. You'll get a similar warning when you try to add a virtual disk and be pointed towards the section to first add a disk.

Then there are times when it'll leave you high and dry. So if you wish to use the devices in a RAID, make sure the disks are formatted as Software RAID, as NAS4Free wouldn't even recognise devices formatted as UFS or FAT32.

RichNAS

Both distros are designed to handle standalone authentication and can also fetch authentication information from a directory server via LDAP. The two distros also support both Linux-style ownership and Windows-style access control lists for fine-grained control. Furthermore, FreeNAS can also be



FreeNAS has a guided wizard to take you through the necessary setup steps.

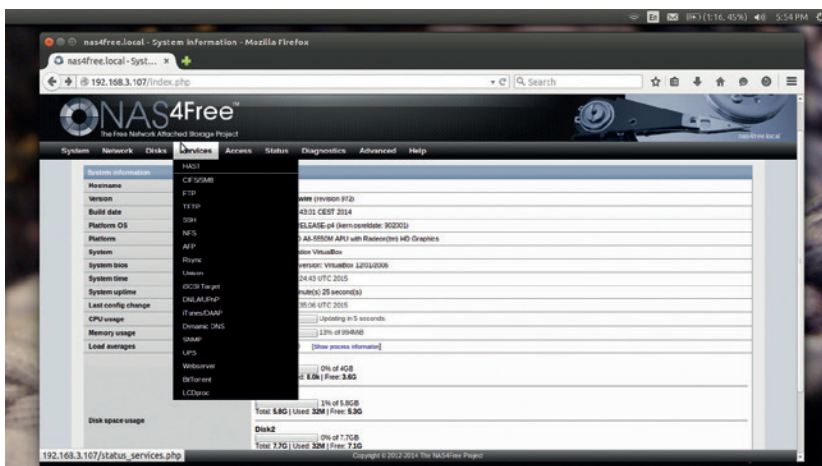
used as an Active Directory domain controller. In fact, some of the most useful NAS features are handled better by FreeNAS. For example, one of the great things about ZFS is its snapshot capability. NAS4Free doesn't offer as many options, nor is it as flexible as FreeNAS when configuring snapshots.

FreeNAS can also be extended with plugins, and uses FreeBSD's Jails mechanism to run them inside isolated silos. This ensures that even if the plugins are compromised they can't affect the NAS. There are plugins

“Some of the most useful NAS features are handled better by FreeNAS.”

that'll convert the NAS into a streaming server and can even run web apps such as OwnCloud. NAS4Free doesn't have plugins as such but has built-in support for several tasks in addition to that of a NAS. You can enable these and use the NAS4Free server as a UPnP Server, Torrent client, DAAP server, and even a simple web server.

Both distros are complex pieces of software, but are well documented, with FreeNAS scoring over NAS4Free in this department as well. FreeNAS also hosts a free webinar daily to help users get started with the distro.



You can easily back up (and restore) the server configuration in an XML file.

VERDICT

<p>NAS4FREE An advanced NAS distro that's designed for advanced users.</p> <p>★★★★★</p>	<p>FREENAS The most feature-rich NAS distribution requires some getting used to.</p> <p>★★★★★</p>
--	--

OUR VERDICT

NAS distros

If we had to award this group test to the distro with the biggest number of features then the top two challengers would have been FreeNAS and its protégée NAS4Free. While both of these solutions pitch themselves to users outside the corporate environment, they'd simply be overkill for most home users. Furthermore, their FreeBSD base and the ZFS filesystem, while a boon to enterprise users, virtually makes them alien technology to the average Linux household.

Instead we'd rather rate the distros based on how they

to make smart choices on your behalf, EasyNAS is a wonderful option. It doesn't have advanced features such as the ability to connect with a directory server, but is very adept at collating disks in a virtual volume or a RAID array. The distro also lets you access data using a variety of protocols including AFP, which lets you use the NAS as a target for Mac OS X's *Time Machine* backup app. The distro masks complexities such as the settings for the various supported protocols and simplifies setting up useful features such as creating snapshots of added

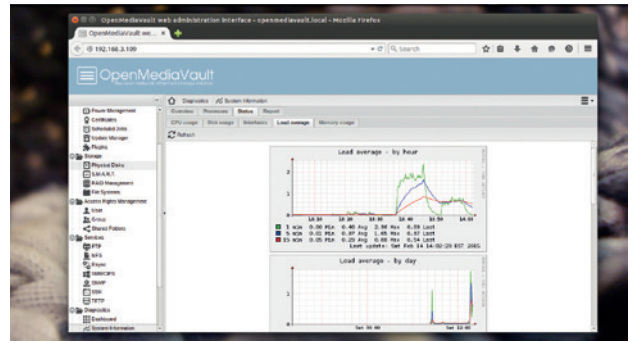
“Open Media Vault is approachable, extremely versatile and our winner.”

manage features with respect to approachability. This is why we rate the Turnkey Linux File Server distro higher than the FreeBSD-based solutions. It the simplest of solutions on test but does what it's supposed to do – provide a network-accessible storage server – without much effort. It ships with a reasonably configured Samba installation and doesn't support any other protocol. You can use the distro to browse its contents from a smartphone, and it can also store backups from any app that supports SMB shares.

But if you need to set up a NAS solution and can trust the system

volumes. Despite its simplicity, EasyNAS lets you control access to the added volumes using Linux-style ownership controls, which makes it an ideal NAS distro for non-technical users.

However, we'll award this group test to the Debian-based Open Media Vault distro. In our opinion it offers the greatest number of features without compromising on usability. The distro has all the commonly-used features you'd expect in a NAS distro and can also be extended easily with dozens of plugins, which makes it approachable, extremely versatile and our winner.



If you want to deploy OMV, check out our tutorial in issue LV009.

1st Open Media Vault

Licence GNU GPL v3 Version 1.9

www.openmediavault.org

Its familiar underpinnings, navigable interface, and adaptability makes it our top choice.

2nd EasyNAS

Licence Several free software licences Version 0.5.3

www.easynas.org

Ideal for extending the benefits of NAS to non-technical users.

3rd Turnkey Linux File Server

Licence GNU GPL Version 13.0

www.turnkeylinux.org/fileserver

Easy peasy network-wide access to data.

4th FreeNAS

Licence BSD Licence Version 9.3

www.freenas.org

Comprehensive solution that'll appeal to enterprise users.

5th NAS4Free

Licence BSD Licence Version 9.3

www.nas4free.org

If FreeNAS doesn't work for you, this just might.

6th Openfiler Community Edition

Licence GNU GPL v2 Version 2.99

www.openfiler.com

A feature-curtailed version that doesn't offer anything worth recommending it over the others.

	Based on	Usability	Data encryption	Plugins	Documentation
OpenmediaVault	Debian	4/5	N	Y	4/5
Openfiler CE	rPath	2/5	N	N	0/5
TL File Server	Debian	3/5	N	N	2/5
EasyNAS	OpenSUSE	4/5	N	N	2/5
FreeNAS	FreeBSD	2/5	Y	Y	5/5
NAS4Free	FreeBSD	2/5	Y	N	4/5

SUBSCRIBE

www.linuxvoice.com/uk-sub



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

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

UK subs prices

12-month print & digital: **£55**
12-month digital only: **£38**

Get 114 pages of tutorials, features, interviews and reviews every month

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

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

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

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



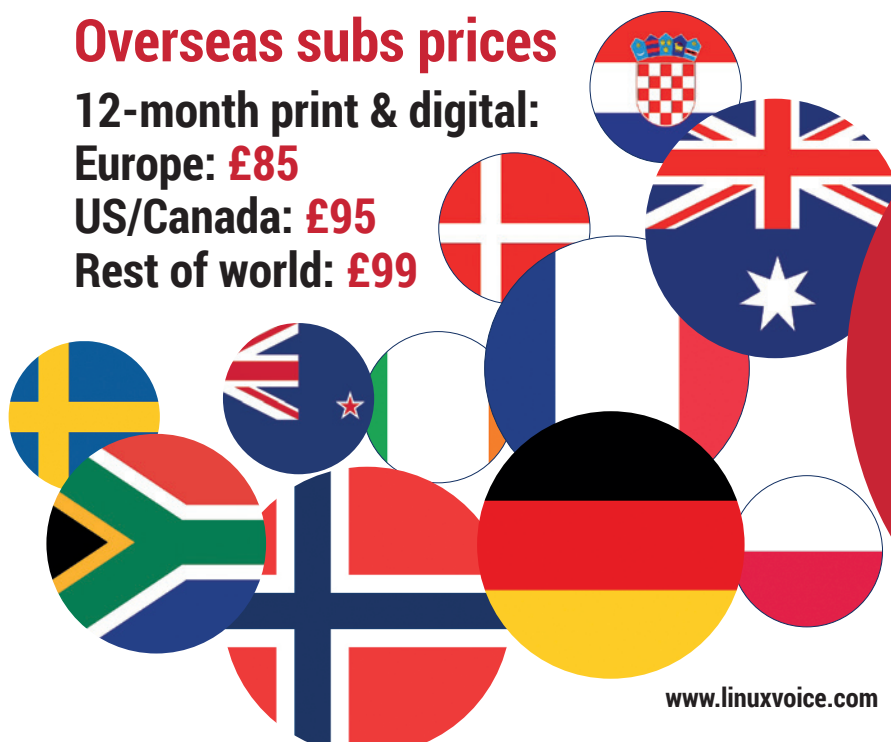
Overseas subs prices

12-month print & digital:

Europe: **£85**

US/Canada: **£95**

Rest of world: **£99**



DIGITAL SUBSCRIPTION*
ONLY £38

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

SYSADMIN: SAVE TIME WITH WEBMIN

Annoyed by niggling differences between Linux distributions? Try Webmin, a consistent web-based tool for system administration.

If you work with multiple Linux distributions, or different flavours of Unix, you've probably come across frustrating little discrepancies between them. Commands often use different flags, filesystem locations can vary, and you often end up hunting through man pages just to get basic jobs done. Some distributions offer text-based or GUI administration tools (such as OpenSUSE's *Yast*) to mitigate this problem somewhat, but you still end up with every Linux OS having its own way to configure things.

Webmin is one solution – and a very fine one. It's a web-based interface for system administration on Linux and other Unix-like systems, providing a consistent way to manage users, run services, configure

firewall settings, set up email, and perform many other tasks. *Webmin* has been in development for almost 20 years, having seen a number of interface revamps along the way, and today it runs on a huge range of platforms – see www.webmin.com/support.html for a list. Along with the usual suspects among Linux distributions, *Webmin* also runs on the various *BSD flavours, IBM AIX, Solaris and other Unix platforms.

You may find *Webmin* in your distribution's package repositories; if not, download the tarball from www.webmin.com (see the TAR link in the top-left), then perform the following commands as root to extract it into `/usr/local` and run the setup script:

```
tar xfv webmin-1.730.tar.gz -C /usr/local
cd /usr/local/webmin-1.730
```

`./setup.sh`

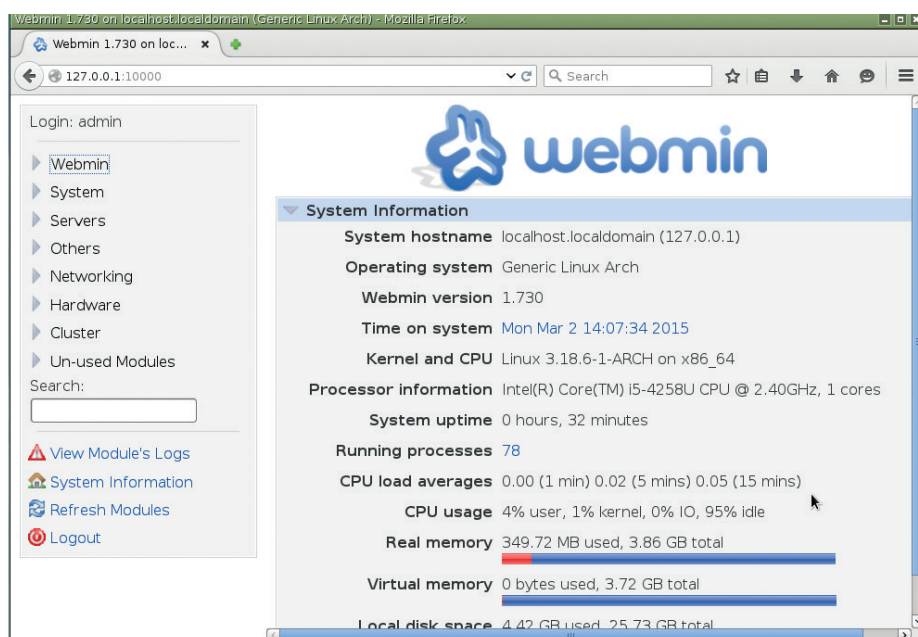
Webmin will ask you where it should save its configuration and log files (the defaults are fine), and then ask you to specify the Linux/Unix system you're using. You'll see a list of many distros – but if yours isn't there, choose 95 for "Generic Linux". In the following step, you'll be asked to configure the web server port, username and password that you'll use to log in. *Webmin* is written in Perl, and if you have the Perl *SSL* library installed, you'll be able to log in over HTTPS.

Finally, the installer will ask you if *Webmin* should be started at system boot, and you'll be able to log in by going to `http://<hostname>:10000` in a web browser (eg `http://127.0.0.1:10000` to administer the local machine). Enter the username and password you specified before, and you'll arrive at the main *Webmin* page, with a panel of system information in the middle and a tree of administration tasks down the left hand side.

You may also see a warning box at the top which states that some modules are out of date. *Webmin* is not a monolithic tool; its functionality is provided in the form of modules. For instance, there's a module for configuring the *Apache* web server, another module for configuring *MySQL*, and so forth. *Webmin* can update these modules with a single click, so it's a good idea to make sure they never get out of date.

Typical tasks

To find out what *Webmin* is capable of, click on System on the left to open the tree of items. You'll see options for changing passwords, managing filesystem partitions, making backups, killing/restarting/reniceing



Webmin's front page shows information about the system, including CPU load average, RAM usage, free disk space and the all-important server uptime duration.

processes, scheduling commands, managing users and groups, and other common tasks. Try clicking some of the options and exploring the panels that come up – they're very useful, even if they don't have checkboxes for every single parameter used by the equivalent command line tools. For day-to-day jobs, though, *Webmin* does a sterling job.

Module configuration

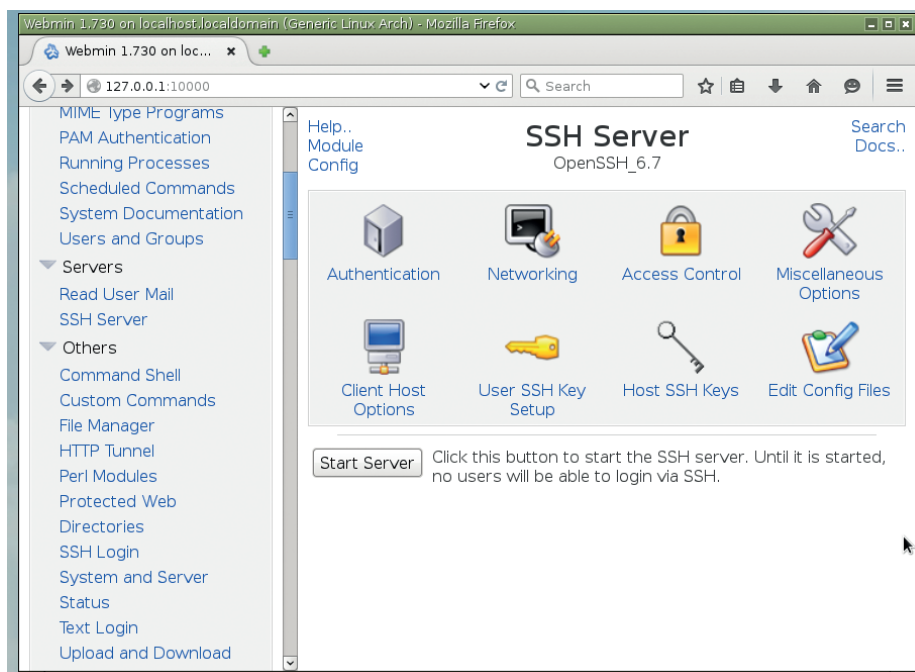
Under the Servers, Networking and Hardware links on the left, you'll find more modules for administering the machine. You'll also see a section called Unused Modules; this contains modules that haven't started because they believe that a certain piece of software isn't installed on your machine. But the detection methods aren't perfect here – for instance, in the case of the *Apache* module. You may have *Apache* installed, but the module complains with “server executable `/usr/local/apache/bin/httpd` does not exist”.

Chances are that your `httpd` binary will be in a different place, so click on the Module Configuration link and change the paths to match your installation. You'll see that *Webmin* can try to guess the location of your `apache2.conf` file, but it's better to specify it directly (it may be `/etc/httpd/conf/httpd.conf` on your system). Click Save at the bottom to reload the module, and all being well you'll now be able to administer *Apache*

Webmin for users

As you explore *Webmin*, you may see some functionality that could be useful for normal users on the system, and not just administrators. The *Webmin* developers have recognised this and created *Usermin*, a stripped-down version of *Webmin* with a handful of modules appropriate for typical users. These include user management tools (eg changing your password), reading mail and performing scheduled commands. The goal of *Usermin* is to provide a friendly configuration panel for user accounts, so that users can change their settings and do simple tasks without requiring command line knowledge.

To enable it, go to Unused Modules in the left-hand tree and choose *Usermin*. You'll see that the module isn't installed on the system, but there's a button to download it. *Usermin* will be automatically extracted, so read the text that's displayed and go to `http://<hostname>:20000` in another tab. There you can log in as a normal user account on the Linux/Unix system, and explore the different options in the left-hand tree. Like *Webmin*, *Usermin* is highly configurable and you can enable and disable modules at will (click the Available Modules button).



Some of the modules, such as for OpenSSH, have a GUI-like approach with chunky icons to click.

– see the Global Configuration tab for the majority of the tweakable settings. Next time you use *Webmin*, the *Apache* module will move from Unused Modules to the Servers section.

Helpfully, most *Webmin* modules also provide direct access to configuration files, so you don't need to SSH in manually if you need to make a quick change. It's also possible to enter single commands under the Others

tree on the left: click on Command Shell, which also provides a history of previously entered commands. Under the Upload and Download section you can transfer files to the machine, which is mightily useful if you can't use SCP. There's also a file manager, although you need Java installed to use it.

An army of admins

If you decide to use *Webmin* in large deployments, you might need to create multiple user accounts so that other administrators can log in. Click the Webmin Users tree icon in the top left and then Webmin Users. Next, click Create A New Webmin User and fill in the details. It's possible to limit access by IP address, and choose specific modules that the user can access. This is important if you want a certain admin to do work on *Apache* and *MySQL*, for instance, but not be able to change anything else on the system. You can also restrict

Webmin users to logging in on certain days, or even certain times of the day.


And there's more: after clicking on a module, go to the View Module's Logs link (it's in the bottom of the left-hand tree) to view recent activity. You can, for instance, see what changes have been made by

Webmin users, so you can point the blame if someone damages the configuration.

Webmin itself is highly customisable.

Go to Webmin >

Webmin Configuration in the left-hand tree menu, and you'll see that you can tweak the appearance, change login settings (such as blocking hosts if they have repeatedly failed to log in) and even upgrade *Webmin* in place. It's also possible to add third-party modules here, many of which you'll find on the *Webmin* website at www.webmin.com/cgi-bin/search_third.cgi?modules=1. If you're familiar with Perl, you can create your own *Webmin* modules by following the guide at http://doxfer.webmin.com/Webmin/Module_Development.

On the whole, *Webmin* is a mature and reliable tool that, when deployed across many different Linux distributions and Unix flavours, can save you a lot of time. Having a single interface for configuration – regardless of operating system – and the ability to provide restricted accounts for other admins will make your life easier when looking after a large number of boxes. 

FOSSpicks

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



Hunting snarks is for amateurs – **Ben Everard** spends his time in the long grass, stalking the hottest, free-est Linux software around.

Webmail client

Roundcube

There are email clients for just about every modern computer system, and most not-so-modern systems. Native email clients work well, but they can be fiddly to configure. This isn't a problem for a machine you use regularly, but if you only use a machine occasionally, it may not be worth it. The alternative is web-based email.

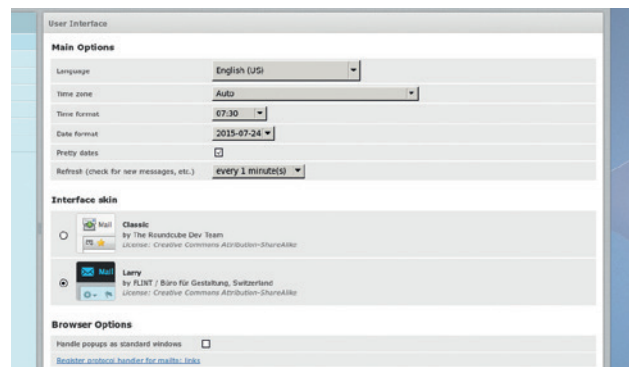
There are loads of services that will host email for you on a website (Gmail, Yahoo Mail, Hotmail, etc), but these are provided by companies that rely on advertising to make money, and we're not that comfortable allowing them access to all our communications.

To solve this conundrum, there's *Roundcube*, a webmail server that you host yourself. You just need to set it running on a machine with a publicly-routable IP address, and you can keep up with your emails

from any machine. We like this solution so much, we use it for our Linux Voice email.

Roundcube is just an email client, so you need to pair it with the appropriate servers for sending and receiving email. Just configure it as you would any desktop email client and you have a webmail system.

We've been running *Roundcube* for a little over a year, and haven't come across any major problems in that time. It's a little more limited than some native clients, but it does have an address book, filters for automatically sorting emails, and it handles HTML without any problems. The only key feature of



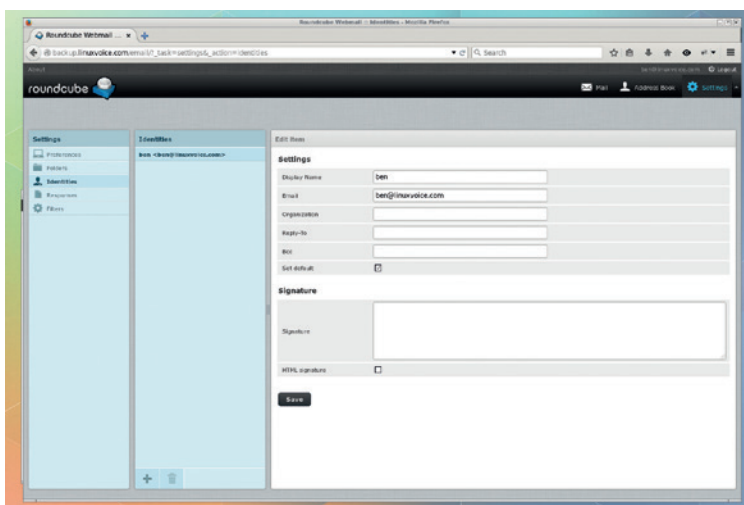
The settings in Roundcube are easy to understand and use.

“At Linux Voice we’ve been running Roundcube as our email client for a little over a year.”

an email client that's missing is PGP/GPG encryption. It is possible to do this with some external tool such as *Mailvelope* (<https://www.mailvelope.com>). In other words, there's nothing missing that we would consider essential in an email client, but power users may find things missing that they like. For example, there's only one type of flagging available.

You can extend *Roundcube* with plugins, and you can see all those available at <http://plugins.roundcube.net>. You can also change the look of the interface through skins. We've never felt the need for either plugins or skins, but if you have a particular need in mind, it's worth checking to see if there are ways to meet this.

The new release brings a few minor improvements (better handling of HTML images and improved searching), but nothing that fundamentally alters this excellent software.



Roundcube supports multiple sending identities allowing you to manage a range of accounts from a single login.

PROJECT WEBSITE
www.roundcube.net

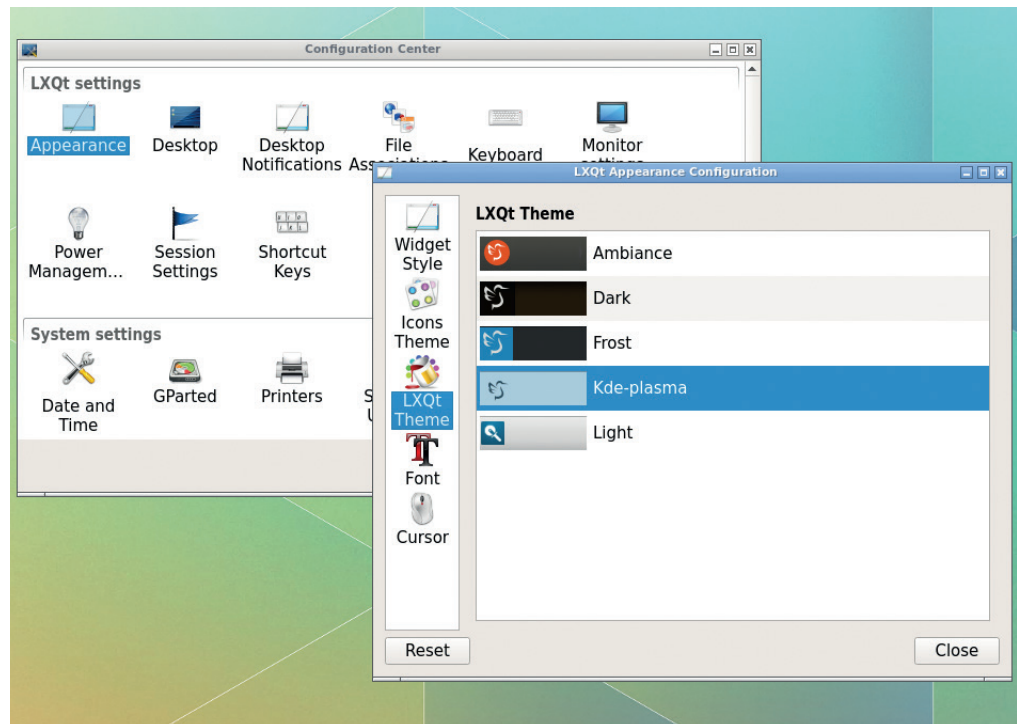
Lightweight desktop environment

LXQt

LXQt is a project created by the developers of the *LXDE* and *RazorQt* desktops coming together to create a new desktop environment based on the *Qt* widgets rather than the ones from *GTK* (as *LXDE* was). *LXQt*'s just reached version 0.9, so it's still in its infancy, but it's rapidly becoming a great option for low-power machines. The aim is to create a simple desktop that just works as users expect it to, rather than one full of configuration options for all kinds of features that few people will ever use.

Version 0.9 is fully migrated to the *Qt 5* toolkit (as used by the KDE desktop) and has dropped support for *Qt 4*, so it takes advantages of all the latest improvements in the platform. It also sees the inclusion of *KWindowSystem* and *KGuiAddons* from KDE. These reuse code from the other major *Qt* desktop rather than reimplementing the same functionality.

We've been using *LXQt* for the last couple of weeks, and we're impressed. It's fast, has what we need, and doesn't get in the way of normal computer use. We also haven't run into any stability problems, so despite its fairly young age, we can recommend it for general use. As a long-term *LXDE* user, this reviewer is considering



The new release of *LXQt* comes with the new Frost theme, and drops some old ones that weren't as popular.

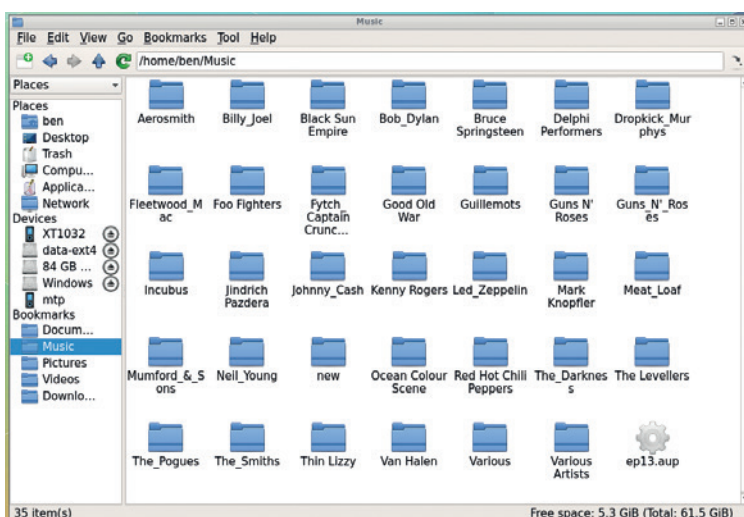
making the switch permanent, and that's about the highest praise you can get from a software reviewer.

While the desktop works well, don't expect too much in terms of

“LXQt is fast, it has what we need, and it doesn't get in the way of normal computer use.”

software. You don't get much more than a desktop and a file manager with *LXQt*. The file manager – a *Qt* port of *PCManFM* – is about as minimalist as it's possible to be, and frankly, we'd prefer a little more functionality. Things like sorting lists of files by clicking on the column header rather than going into a menu, for example, would make the file manager much more pleasant to use.

Much *Qt* software is designed to work with the much more heavyweight KDE desktop; conversely, a lot of the more lightweight Linux software uses the *GTK* widget set. This means that there's not a lot of software that fits the *LXQt* philosophy. This is sure to change though. In fact, it's already starting to: *QTerminal* is a lightweight terminal for *Qt* that fits in well with the *LXQt* philosophy. Watch this space...



PCManFM is too stripped down for our tastes – we prefer using a different file manager with *LXQt*.

PROJECT WEBSITE
www.lxqt.org

Real time OS

NuttX

Compared to most modern operating systems, Linux is quite lightweight, and it happily runs on hardware that wouldn't even boot the latest version of Windows. However, sometimes even Linux is too heavy. NuttX is a very lightweight real time operating system released under a BSD licence, and rather than more heavyweight options, it's designed with tiny processors in mind.

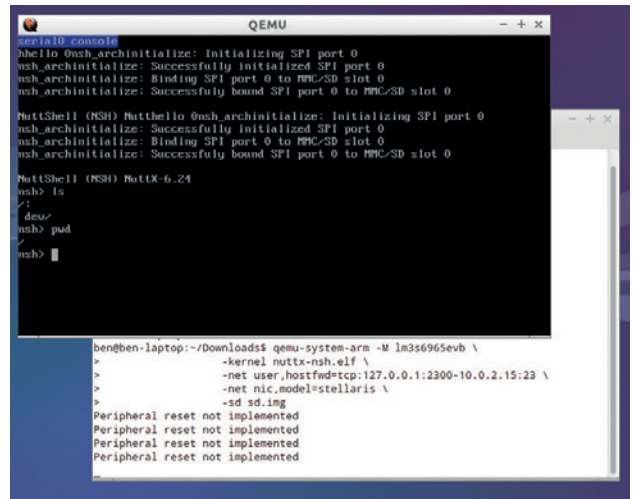
With a bit of persuading, it will run on a Z80 (the processor at the heart of the *Pacman* arcade machine, the Nintendo Game Boy, and many other iconic 80s and 90s computers), an AVR (similar to the chips in most of the Arduino microcontroller boards) and ARM Cortex-M series processors including the Arduino Due. If you like the sound of NuttX, but don't have access to any of these bits of

hardware, you can get started by emulating one using *Qemu*. There are instructions at www.zilogic.com/blog/tutorial-nuttX.html.

The point of NuttX isn't to turn these machines into desktops, but for building embedded devices. As such, it's stripped down to just the barest essentials, but it still strives for Posix and ANSI compliance. It also has a few concessions to ease of use, such as a graphical widgets toolkit (*NxWidgets*) and a shell (*NuttShell*).

Real-time for hardware

This isn't something that you're going to replace your Linux install with, but if you're looking for a high degree of control over hardware, the real-time aspect of NuttOS is quite appealing, especially given that the real time patches for Linux are slowly sliding out of usability.



NuttX boots to *NuttShell*, which accepts some of the same commands as a Linux system.

You won't find much in the way of drivers for consumer hardware, but there's support for peripherals such as USB hosts, flash memory, PWM drivers, CAN buses, DACs, ADCs, etc. If you're building your own hardware, this should be enough to run what you need.

PROJECT WEBSITE
www.nuttX.org

Video compressor

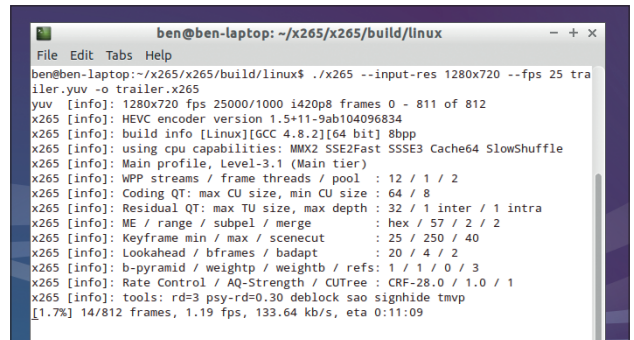
x265

Videos take up huge amounts of space and network bandwidth, and there's a constant effort to improve compression. Almost all video encoding – H.265 included – is lossy. This means you can make the file as small as you like, but as you make it smaller, the quality gets worse. The improved compression in this format means you can either store a video at the same quality in a smaller amount of disk space, or if you're not storing video but streaming it, you can stream higher quality over the same bandwidth or the same quality over a lower bandwidth.

x265 is an implementation of the new H.265 standard (also known as High Efficiency Video Coding or HEVC). This is the latest in a long line of standards from the Motion

Pictures Expert Group (MPEG) and the International Telegraph Union (ITU). H.265 can roughly double the compression ratio of H.264 videos (also known and MPEG4 Part 10 or Advanced Video Coding).

It achieves the compression improvement using many optimisations, but one of the biggest is the use of coding tree units. All video compression splits each frame up into smaller sections and compresses them separately. Previous standards had split the image up into a regular grid, but HEVC varies the size of the sections it splits the image up into. Large sections are used to compress



The increased compression comes at the expense of slower processing. Our test machine could process just one frame per second.

areas with little detail, while smaller areas hold more detail.

x265 is open source, however it's not patent-free. This is a thorny subject, and if you're including x265 in any form of software, you should make sure you fully understand the situation. However, until Dalla – the patent-free codec developed by Mozilla and Xiph – is ready, there aren't any modern patent-free alternatives.

“x265 improves compression using many optimisations.”

PROJECT WEBSITE
<http://x265.org/>

File manager

Worker

All desktop environments come with a file manager, but you may not always find that the default one meets all your needs.

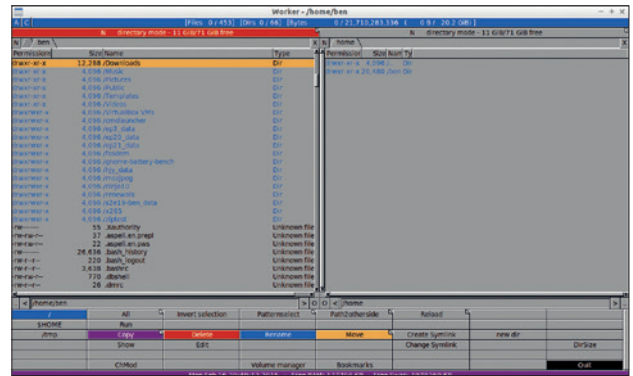
Worker is a two-pane file manager. This style of file tool is less popular than it used to be, but still a very effective method of dealing with your files. Each pane is independent of the other, and you can browse different directories with each one. The big advantage of two-pane file managers is when you come to move files around. You can have one pane in the directory they're coming from, and the other in the directory they're going to, and then you can shuffle them around without the risk of them going missing. By far the most famous two-pane manager is *Midnight Commander*, which is a fantastically useful tool, but the terminal

interface won't suit everyone. *Worker* brings the same level of power to the GUI world.

It's more than just two panes, though. It can also be used with AVFS (a virtual filesystem) to browse the contents of many compressed files. It has labels, bookmarks and a file search tool for finding what you need, and can use external programs to perform any missing functionality.

The look could either be described as retro or awful depending on your feelings about user interfaces. This look is due to it being built directly in X Windows and not using a widget toolkit. While this doesn't make for

“Worker’s look could either be described as retro or awful.”



The garish colour scheme won't be for everyone, but at least it makes it easy to see what's going on.

particularly stylish graphics, it does mean the code is very lightweight and extremely portable (if X runs on the system, then *Worker* will).

The target user of two-pane file managers, unsurprisingly, is system administrators who deal with a lot of files, but they are useful for anyone who has to move loads of files from one place to another.

PROJECT WEBSITE
www.boomerangworld.de/cms/worker

Exercise database

SportsTracker

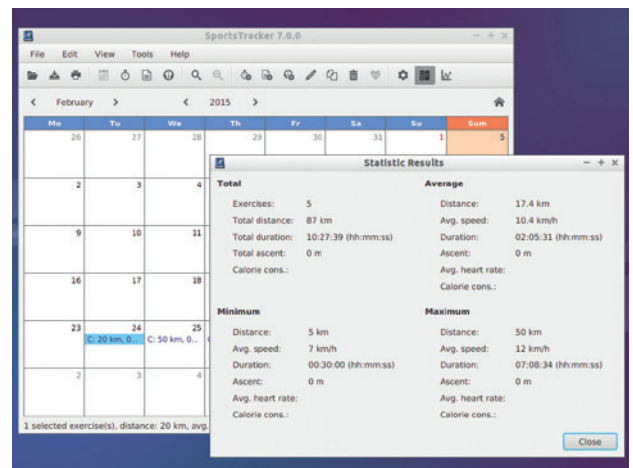
There are plenty of free-to-use commercial tools for tracking how much exercise you're doing. However, this is one area that we think that free as in free software is really important. Not only is this key to health, but it's an area in which you'll build up a vast amount of data over the years. Commercial tools may monetise this data in various ways, and the companies that run them don't have your best interests at heart.

Plus, with a free (as in free software) tool, you know that you can get the data back if you need – although *SportsTracker* doesn't directly support exporting to formats suitable for other software, the open nature of the program means you can create a filter for the file format should you need to. Freedom aside, *SportsTracker* does

its job well. It can import data from a wide variety of sources (including GPX, so you can get data from GPS receivers on mobile phones). It doesn't support a wide variety of sports types – just running and cycling – so depending on what you do, it may not be suitable for you. It also doesn't try to calculate the number of calories consumed, but the estimates that some applications make in this area are highly inaccurate anyway. As well as exercise, you can track your weight, so you can see if the workouts are having the desired effect, and you can plot how quickly you're losing weight.

Running free

SportsTracker 7 is newly re-written with a JavaFX interface (rather than Swing, which was used previously).



The statistics screen gives you an overview of the exercise you've been performing over any given timespan.

It looks clean, and should run on anything with a Java runtime.

Commercial tools may have more functionality, but the knowledge that we're not entrusting our personal health data with third parties who are free to sell it on is a winning feature.

PROJECT WEBSITE
www.saring.de/sportstracker

Secure deleter

srm

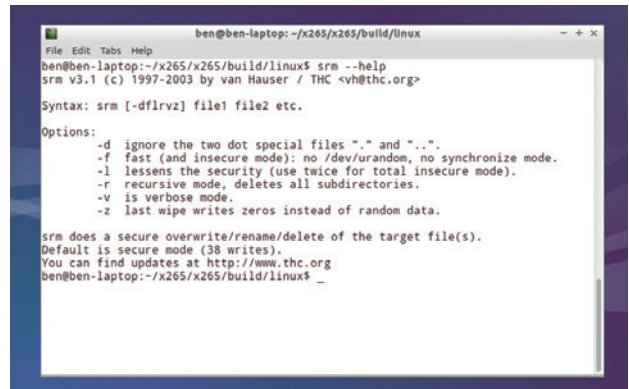
The **rm** (remove) command is probably one of the best known Linux tools. It's a simple command line utility that lets you delete files, and when combined with a powerful shell, it can be used to easily delete a range of files that satisfy particular criteria. However, it is badly named. It doesn't actually remove the files at all; it just deletes the reference to them. This means that if you delete a file using **rm**, the data is still there.

If you're just deleting a file to free up some space, then this isn't a problem. However, if you're deleting a file because it contains data you don't want to be on a machine any more, then **rm** isn't sufficient. You also need to remove the data.

Secure Remove (**srm**) is one solution to this. It's a drop-in replacement for **rm** that not only deletes the file in the same way **rm**

does, but it scrubs off the data in a way that would please even the most paranoid people. First, it overwrites the data with all 1's. Then it overwrites it with random data – five times – then it overwrites that random data with special patterns that are designed to make it impossible to recover the data using even the most sophisticated theoretical recovery techniques – 27 times – then another five overwrites using random data. Only after all this overwriting does it consider the file blanked, and then it deletes the file.

The result is that there's no way that anyone – not even the most powerful state-sponsored attackers



A full 38 wipes is probably overkill, but it's better to be safe when deleting sensitive data.

– can recover the data if the data's stored on a magnetic hard drive. Solid state drives can pose problems and are much harder to guarantee secure deletion. The downside of **srm** is that it takes longer to perform than normal **rm**. Large files can take several minutes to blank. That's the price of security.

“srm scrubs data in a way that would please even the paranoid.”

PROJECT WEBSITE
<http://srm.sourceforge.net>

Finance manager

HomeBank

In today's world of online banking, where smartphone apps can tell you your bank balance, it's easy to keep track of how much money you have. However, it can be tricky to find out where all of your money is going. That's what *HomeBank* is for. It can import the QIF and OFX files used by online banks as an export format (note – QIF files didn't work in our testing, but this could have been a problem with our bank's export function). *HomeBank* then enables you to tag each transaction with what type it is. Using this, you can get a detailed picture of how much you spend on what.

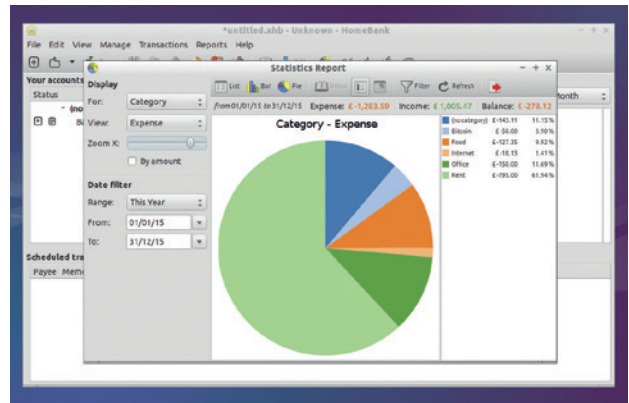
You can take this simple method further by assigning a budget to each category of expense, and then see how you're performing with respect to these targets. You can

then track these budgets over time, and make sure that you're realistic about your spending habits.

Annual expenditure £19 0s 6d

How well *HomeBank* will work for you will depend on how you spend money. If most of your expenses are on a card, then it's easy to import everything into the application. However, if most expenses are cash then they will each have to be created manually, and it might be easier to use a spreadsheet or some other custom tool. We found that we could go from an exported file from the bank to a detailed breakdown of expenditure very quickly.

Finance software can sometimes be intimidating, but *HomeBank* is easy to use, even for people new to finance. You don't need to know



HomeBank can display data in pie, bar and line charts, giving you three ways to view the vultures pecking at your hard-earned money.

any fancy jargon, or wade through archaic forms. It all just works as expected. Should you find yourself stuck though, *HomeBank* is well documented at <http://homebank.free.fr/help/index.html>. If you're unsure where your salary goes at the end of each month, then this is the software for you.

PROJECT WEBSITE
<http://homebank.free.fr>

FOSSPICKS Brain Relaxers

Ball balancing challenge

Neverball

Using the mouse or keyboard, you control the tilt of a board. On that board is a ball, and you have to use the tilt to manoeuvre the ball around various mazes, collecting coins as you go, until you reach the finish. It sounds simple, and in many ways it is. The enemy here isn't complex AI, or other players, but simple physics. The faster you move through the maze, the more momentum the ball has, and the harder it is to control. Slow and steady or fast and reckless – which is better? We chose the latter option, but then here at Linux Voice Towers, we're used to life on the edge.

This game doesn't stand out only for its gameplay though. It's also the first desktop Linux app to

work with Gnome's sandboxing applications project. This project uses a group of nifty technologies to package applications up in such a way that they are distro-independent, and can be limited in how they interact with the system in order to improve security. This is similar to how apps work on Android. There's a blog post by Alexander Larson about the process of sandboxing *NeverBall* at <http://blogs.gnome.org/alexl/2015/02/17/first-fully-sandboxed-linux-desktop-app>.

It's early days for sandboxing on Linux (at least, non-Android Linux), so we can't say for sure how well it will end up, but we're cautiously optimistic about the potential for more secure and portable (between distributions) packaging. The early



The 3D graphics are simple enough to run on most computers, but rich enough to provide an immersive experience.

indications are that it will make it far easier to release software for Linux, and that's got to be a good thing. In the mean time, you should also find *Neverball* as a regular package in your distro's repositories.

PROJECT WEBSITE
<http://neverball.org>

Transport simulator

OpenTTD 1.5

OpenTTD is based on the classic 90s game *Transport Tycoon Deluxe*.

The aim is to build a thriving, transport business by connecting a map with transport links, and plying the routes with vehicles.

The gameplay is a little confusing to start with, so it's probably best to start with the tutorials at <http://wiki.openttd.org/Tutorial>, especially if you're not familiar with the original game. This will take you through building the infrastructure, buying vehicles, and putting them to work on routes. All this is needed before you can start to make money off your fledgling transport empire. Once you've learned the basics, the learning

curve gets a bit shallower, and it's easy to progress through the game using more and more advanced transport options without resorting to the documentation too much.

OpenTTD contains a number of improvements including larger maps, an online multiplayer mode and more advanced transport options. These make the game far richer and allow enjoyment for longer than the original.

Despite all the enhancements, *OpenTTD* is true to the spirit of the original, both in gameplay and in style. The isometric view and



You can use land, sea and air to move cargo and people as efficiently – and as cost-effectively – as you like.

pixelated graphics will provide a welcome dose of nostalgia to anyone who enjoyed PC gaming in the 90s. Play it on a CRT screen, put some Oasis or Blur on the stereo and hunker down for some retro fun. 🎮

“OpenTTD’s improvements make it far richer than the original.”

PROJECT WEBSITE
www.openttd.org

SUBSCRIBE

www.linuxvoice.com/us-subs



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

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

US/Canada subs prices

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

Get 114 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
30 APRIL

COMPLETE GUIDE TO HACKING

ETHICAL HACKING*

Learn how the bad guys work and use that knowledge to protect yourself. Starring Ben Everard and the Metasploit framework.

*We know we promised this before, but it's really happening this time.

EVEN MORE AWESOME!



Larry Wall

As the creator of the Perl programming language, this man practically wrote the internet single-handedly. Probe his mind and enjoy his taste in shirts!

CC BY-SA Klappi

These are unscaled parentheses:

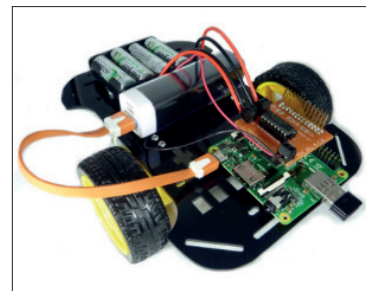
$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$$

And these are scaled:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$$

Lyx

If you like nicely formatted scientific documents (and who doesn't?) but don't want to bend your brain learning *Latex*, try its WYSIWYG cousin, *Lyx*.



Raspberry Pi robots

The first rule of robot club is that robots should be cheap to build and easy to program – which is why there are so many kits powered by the Raspberry Pi...

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

Copyright Linux is a trademark of Linus Torvalds, and is used with permission. Anything in this magazine may not be reproduced without permission of the editor, until December 2015 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2014
ISSN 2054-3778

Subscribe: shop.linuxvoice.com

DATES FOR YOUR DIARY

DEVOPS Spring 2015 • 24th, 25th & 26th March 2015
The Hilton York, 1 Towers Street, York YO1 9WD

Tuesday 24th March - Workshops

'Zero to Perl' half-day - provided by Shadowcat Systems Ltd.
'Practical Digital Forensics' half-day - tutor: Tim Fletcher
...Further Workshops to be Announced...

Wednesday 25th & Thursday 26th - Conference

provisional list of accepted talks to date:

Julien Pivotto - *Shipping your product with Puppet code*
Stephen Quinney - *Intrusion Detection using the Linux Audit System*
Toshaan Bharvani - *Virtual Machine Lifecycle Management with Anisble*
Kenneth MacDonald - *Kerberos - Protocol and Practice*
Wim Godden - *Intrusion detection through backups (and other security tricks)*
Pieter Baele - *Linux centralized identity and authentication interoperability with AD*
Stu Teasdale - *Beyond Blue-Freen: Migrating a legacy application to CI and the Cloud*

2nd Quadrant - *State of PostgreSQL Database 2015*
Bernd Erk - *Open source Monitoring with Icinga*
Mark Cairney - *Federated Access Management*
Peter Tribble - *Creating Tribblix*
Richard Melville - *An Introduction to Btrfs*
Nick Moriarty - *Puppet as a Legacy system*
Mark Keating - *Who's The Pan* plus many more...

for more information see: <http://www.flossuk.org/Events/Spring2015>

Book your place now! Use code 'VOICE15' for a 15% discount on Conference fees.

This event is sponsored by: ELIGO Recruitment

OpenTech 2015 • Saturday 13th June 2015
ULU, University of London Union

OpenTech will be 10 years old in 2015.

This event is only possible because of wonderful and generous sponsors in the past.... If you or your company is interested in Sponsoring please get in touch - opentech@opentech.org.uk

The event's predecessors were low cost, one-day conferences about technologies that anyone can have a go at, from 'Open Source' style ways of working to repurposing everyday electronics hardware.

<http://www.opentech.org.uk>

Dynamic Languages Conference - Saturday 20th June 2015 - Manchester
for more information see: <http://www.dynamiclanguages.co.uk>

If you are not a member and want to be kept informed of future events please subscribe to ['announce@ukuug.org'](mailto:announce@ukuug.org) see: <http://lists.ukuug.org/mailman/listinfo/announce>

UKUUG - Bronze Sponsor
members include:



Why not join today? Annual Individual membership is just £42.00 inc. VAT. See: <http://www.flossuk.org/join>

As a member you can attend all our events at the specially discounted member rates and receive our quarterly Newsletter!

UKUUG Ltd. t/a FLOSS UK, The Manor House, Buntingford, Herts SG9 9AB
Tel: 01763 273475 Email: office@ukuug.org



TUTORIALS

Dip your toe into a pool full of Linux knowledge with nine tutorials lovingly crafted to expand your Linux consciousness



Ben Everard

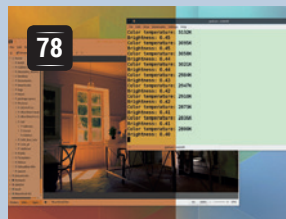
is combining all this issue's tutorials into a super project that will alter computing for ever.

The standard iOS-alike interface for smartphones is the same on almost every phone platform and hasn't really changed in five years. Back then, phone processors were much less powerful and screens much lower resolution. The problem is that now, people are used to the way a particular system works, and it will be hard for Android or iOS to change. Most of the newer phone OSes have aimed for similarity rather than innovation. The only company daring to think differently on this front is Canonical, and I for one applaud its efforts. Creating an entirely new ecosystem is a risky strategy, but then so is every attempt to enter the mobile phone business.

I don't think anyone can honestly claim to be able to predict the future of an industry as fickle as the mobile computing industry. My guess is that the next few years will see an end to the Android-iOS duopoly as several of the new OSes gain traction. If this duopoly is broken, the ability to run on multiple platforms will become an important feature in apps. If customers start demanding this, app vendors will comply and it'll become easier to run whatever OS you want, and not be tied into a particular platform's software. If this happens, Canonical's risky move might just pay off.

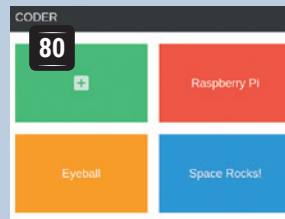
ben@linuxvoice.com

In this issue...



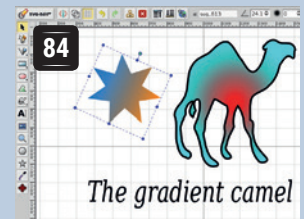
Redshift

Graham Morrison alters the colour balance of his screen to keep his eyes fresh for long evenings spent playing computer games.



Google Coder

Want to get started with programming web technologies? **Les Pounder** introduces a programming tool from Google.



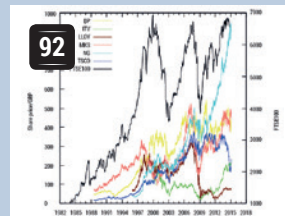
Vector Graphics

Marco Fioretti shows you how to make high quality, scalable images that will make your website look great on any screen.



x86 CPUs

Ever wondered what goes on inside your computer's brain? **Valentine Sinitsyn** takes you on a tour of your CPU to find out.



Share Trading

With a bit of Java, you can start a trading empire. **Andrew Conway** shows you how to build a fortune on the stock exchange.



Mail Server

John Lane keeps his data secure by running his own email server using *Roundcube* and *Cyrus*. You can too!

PROGRAMMING

Fortran

100 This language dates back to the 1950s, but it's still with use today. This long heritage makes it the perfect subject for our first look back at the history of programming languages. Join us as we dust off the history books and head back into a world before syntax highlighting, code completion and Stack Overflow.

Packaging

104 So, you've created some wonderful code that will transform the future of humanity. That's great, but how do you get it out to the world? In this article we look at Python's options for sharing code through modules and packages, and how to make the most of them to get your code out to a wide audience.

ASM

106 In part three of this series, **Mike Saunders** reveals his sadistic side and starts programming without the advantages of even an operating system. Here, you'll learn how to use just the BIOS and assembly language to power his programs. Only the truly geeky should risk following this path.

EASE EYE STRAIN AND SLEEP EASIER WITH REDSHIFT

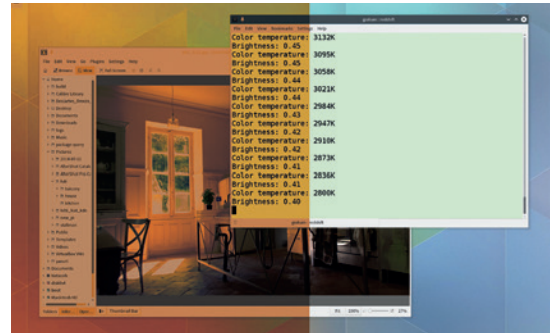
Dynamically adjust the whiteness of your screen to reduce eye fatigue

WHY DO THIS?

- Sleep better.
- See better.
- It's an excuse to revel in the brilliance of Lord Kelvin.

Too many of us work late into the night staring at a screen. And while the best solution is always going to be to work less, if you haven't got that luxury Free Software can offer you the next best solution – a tool that adjusts the whiteness of your screen as day turns into evening and evening turns into night.

Not only will this reduce the fatigue on your eyes, it also helps reduce the stress on your precious neurons, helping tell your brain it's not really midday but nearly time for bed. If that last sentence sounds more like alternative therapy, all we can suggest is that you try it for yourself, because it's brilliant.



Redshift changes the white balance of your screen so that it's easier on your eyes.

Step by step: Grab and configure Redshift

1 What Redshift does

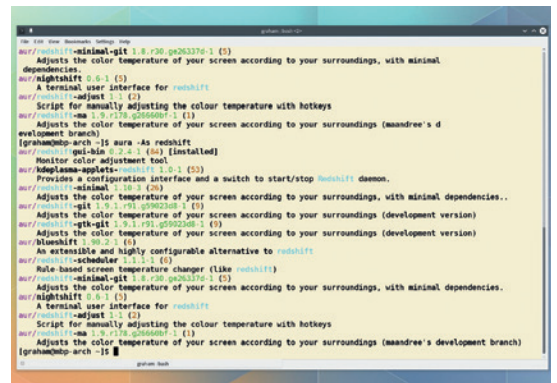
Redshift is an open source tool that dynamically adjusts the white balance of your display over time. If you're a photographer, you'll already know what white balance is – it's the process of adjusting the colour balance of a photo to ensure white is as neutral as possible, because the way white appears changes under different lighting situations. You don't notice these changes because the brain automatically compensates for lighting conditions. It will keep on telling you something is white whether it's lit by the midday sun or by a late evening sunset.

It's only when you take a photo and look at that image under different lighting conditions that you might notice. Photographers take something they know is white from the photo and adjust the entire colour balance until it is. Redshift does the opposite, changing the colour balance of your screen as if it were lit from a different source.

2 Install redshift

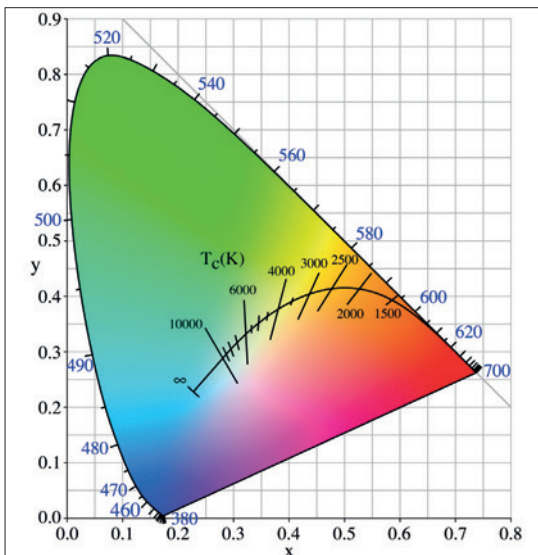
This may sound like Redshift is reducing the overall quality of your display but all it's really doing is adjusting the white balance so that white now looks like it's lit by the sun at the current time and date and also geographical location, or if it's dark, the sun is replaced by lamp light. This is both easier on your eyes and helps trick your brain into preparing itself for the appropriate time of the day.

Redshift has become popular enough to have spawned several side-projects. It is itself the open source equivalent of a proprietary utility called *f.lux*, and you'll need to avoid these when installing these from your distribution's package manager. We're going to stick with the simple **redshift** package to get things started and to explain some of its options; you can then go back and explore some of the alternatives if you like what it does.



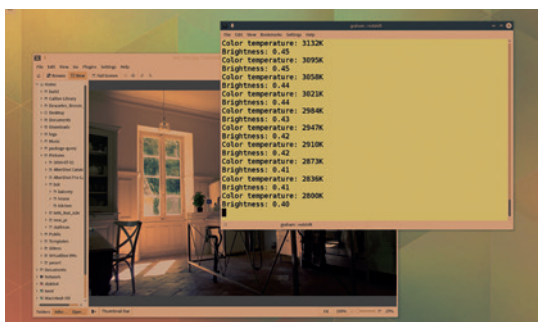
3 Colour temperature

With *Redshift* installed, execute it from the command line by typing **redshift -O 3000**. The 3000 is using a unit of measurement for temperature called Kelvin, also used for white balance and colour temperature within screens, because black bodies (originally carbon in William Kelvin's experiments) emitting heat at around 3,000 K look orange, whilst those of around 8,000 K look blue (as shown in the image – the black line is the change in colour as K increases). A neutral colour is considered to have a Kelvin value of 6500, and a candle burns at around 1900 K.



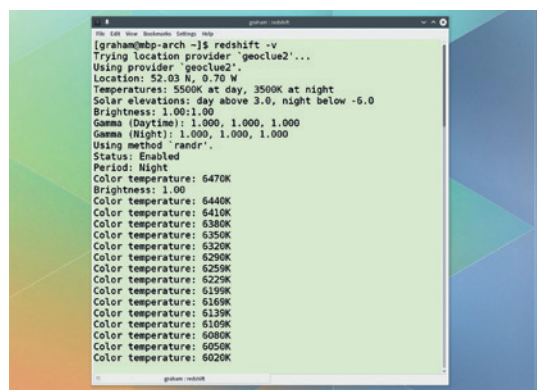
5 Customise Kelvin and brightness levels

If GeoClue has any difficulty finding your location, you can manually enter your latitude and longitude by using the **-l LAT:LONG** argument. There are lots of online services that will take a postcode and turn it into your location. Another important option is the ability to change the colour temperatures your screen is going to shift between. This uses the argument **-t**, and we prefer a more extreme night value of 2800 K, which you can pass to *Redshift* with **redshift -v -t 6500:2800**. It's worth looking up the Kelvin values for other kinds of lighting. Another additional argument is brightness adjustment. This isn't the same as hardware brightness, and won't really extend your battery life, but it gives you more granular control over your screen in low light. The argument for this is **-b DAY:NIGHT**, where day and night are values between 1.0 and 0.1.



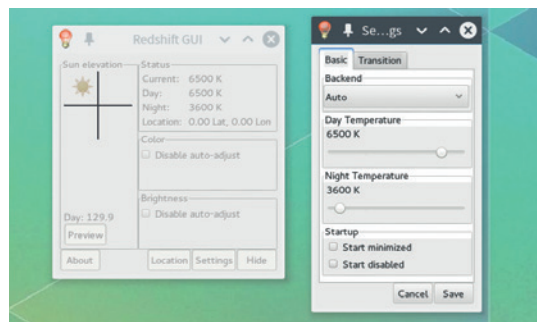
4 Geolocation

Without any arguments, running **redshift** will attempt to detect your geographical location automatically using GeoClue, a D-BUS service that uses your network connections to determine your location locally. This is so it can adjust the colour temperature against the respective location of the sun at your latitude and longitude. After detecting your location, your screen's colour temperature will adjust gradually moving between 3500 K for night and 6500 K in the day. If you add the **-v** option you can see how *Redshift* is changing, but after the initial transition, changes should happen so gradually that they're not noticeable, so it's only when you really do look at 6500 K light, such as a backlit keyboard or white LED, you'll realise your eyes and brain had adjusted to the new status quo.



6 Use a GUI

Even though *Redshift* runs perfectly from the command line, and we'd recommend launching it and forgetting about it, there are numerous interfaces to its various functions. These can give you better control over the colours it produces and the times it produces them. *Redshift-gtk*, for example, adds an applet widget to remind you it's running. Another option is *Redshift-gui*. This gives you a graphical indication of the sun's position, lets you set a location, and fine-tune colour temperatures and transition speed – basically all the options you get from the command line, only from the convenience of your mouse. And whatever option you choose, typing **redshift -x** will always reset your screen to its default values. *Redshift* is one of the best utilities we've ever used. When you turn it off and look at real white again, you can't believe your eyes. 📺





HTML, CSS AND JAVASCRIPT ON THE RASPBERRY PI

LES POUNDER

Turn your Raspberry Pi into fully fledged web development environment with a little help from Google Coder.

WHY DO THIS?

The internet extends into every facet of our lives and learning how a web page is constructed is a great skill for children to learn. In this tutorial we will use free software from Google called *Coder* to learn HTML, CSS and JavaScript.

TOOLS REQUIRED

- A Raspberry Pi.
- Ethernet or Wi-Fi.
- A computer on the same network as the Raspberry Pi.

When we think of the Raspberry Pi we instantly think of great projects using Python, Scratch and Sonic Pi. But there are also many other languages that can be used with the credit-card sized computer. Three of these languages are HTML, CSS and JavaScript, which together provide a powerful framework for creating web content.

- **HTML** Hyper Text Markup Language is the most common language used to create web pages. It uses a series of 'tags' that identify elements on a page, for example a title, image or video. HTML is not a programming language – it's more of a content/markup language.
- **CSS** Cascading Style Sheets are used to change the look and feel of a web page. A whole site can be linked to just one stylesheet.
- **JavaScript** is a programming language that has matured with use on the internet. It can be used to link HTML forms to MySQL databases or used with a microcontroller to power hardware projects such as the Espruino Pico board.

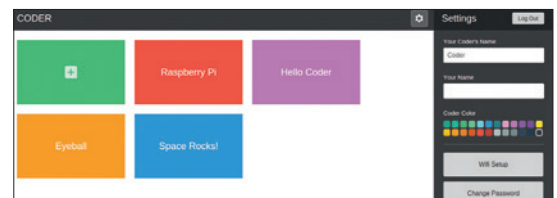
Thanks to the cost-effective Raspberry Pi we can easily create a web development suite using a Linux distribution created by a team of Google employees, this project is called Coder. Coder is an open source operating system that creates a suite of tools to edit HTML, CSS and JavaScript in your web browser. Using Coder you can easily create web apps that are hosted on your Raspberry Pi.

Getting started

Installing Google Coder requires downloading a zip archive from <http://googlecreativelab.github.io/coder/#download> which contains **raspi.img**, an image of a full operating system which is to be copied to an SD card. Extract the archive to a suitable location, and then open a terminal and navigate to the location of **raspi.img**:

```
cd /home/les/Downloads
```

We are going to be using a command called **dd** to



From the Coder main menu you can access the settings menu via the cog icon in the top-right of the screen.

copy the contents on the image file to a blank SD card. In this tutorial we're using Linux Mint 17, but if you are using a Windows or Apple computer, follow the instructions at <http://googlecreativelab.github.io/coder>. The **dd** command is not to be used lightly, as it has the capacity to cause damage if used incorrectly. The **dd** command works as follows:

```
sudo dd if=/location of image of=/location of SD card bs=4M
```

Firstly you will notice that **dd** is preceded by the **sudo** command – this is a safety precaution requiring you to enter your root password.

if refers to the input file, which in our case is the **raspi.img** file. **of** refers to the device that will receive the stream of data, which is typically **/dev/name of SD card**. Lastly **bs** refers to the block size, used to copy a certain amount of data in one block, in this case 4MB.

So we know the location of the **raspi.img** file, but where is our SD card? To find out, insert a blank SD card of greater than 4GB in size into your computer. In the terminal type in the command:

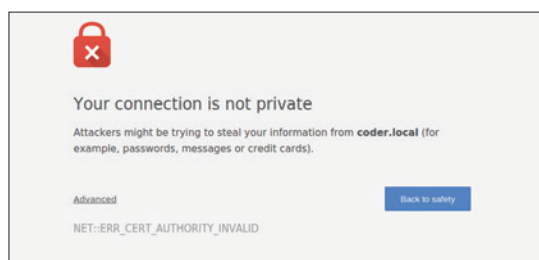
```
mount
```

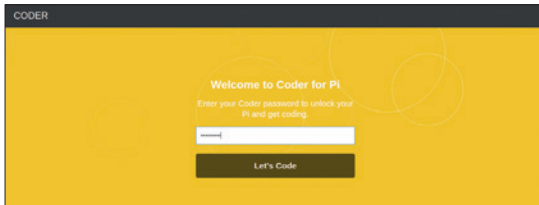
You will now see a list of all the hard drives, USB flash disks and SD cards inserted into your computer. One of those will correspond to your SD card – in our case it was **/dev/mmcblk0**, which is what we need for the **dd** command. So repeat the above **dd** command and substitute the **if** and **of** values for your locations. This command may take some time to complete, so now is a great time to get a cup of tea.

With the image copied to your SD card, unmount the card, and when prompted remove the card from your computer. Now insert the card into your Raspberry Pi, and then insert an Ethernet cable, connected to your router, and finally insert the power adaptor and power up your Raspberry Pi.

With our Raspberry Pi booted we now have an effective web development environment for less than £30, but now we need to access it. On our Linux

When connecting to Coder your browser will warn you that the connection is untrusted; normally this is good advice, but for Google Coder we can trust the connection, so click on Advanced to progress.





After you've created a strong password, Google Coder will ask you to log in using that password.

Mint computer we opened the Google Chrome web browser and navigated to:

<http://coder.local>

When connecting to Google Coder for the first time, you'll be prompted to create a secure password made up of letters and numbers. On the next screen, enter your new password and click on Let's Code.

Once logged in to Coder, you will see a short introduction to the user interface. The green box enables you to create a new application, and the other coloured blocks are pre-made applications that can be explored. In the top-right of the screen is the Google Coder settings menu.

Our project

We're going to create a simple website to learn more about HTML, CSS and JavaScript. And of course we are going to use the Raspberry Pi 2, the latest model from the Raspberry Pi Foundation, as our subject.

To create a new project, click on the green box, and you will be prompted to name your project. You can also select the colour of the box; we chose a fetching Raspberry colour. Choose a colour and then click on Create to continue.

We are now taken to our web application, and we can see many tabs at the top of the screen, the first of which is HTML. We can see that there is already some code in there; leave it there for now. In the CSS and JS (JavaScript) tabs we can also see example code which for the time being can be left as is.

The next few tabs are identified via icons, the first of which is a folder icon. This is the media menu and we can use it to import pictures, files, videos and audio into our projects. The next icon is an eye, which enables you to have a split screen preview of your work in code and the finished results. Our final icon is a gear, which denotes that it controls the settings for our project – we can rename, add an author and change the colour of the project for the main menu.

HTML

HTML is not a programming language; it's a markup language used to position elements on a page. It does have its own syntax, and elements are constructed inside of tags that are encapsulated in "<..>" brackets. An HTML document is constructed like this:

```
<html>
<head>
<!--In the head we store links to external resources such as
JavaScript and CSS.-->
```

Great resources

Google Coder is a great way to learn web development. It comes with a great suite of tools to enable you, but if you need a little theory to help you understand the practice then there are plenty of great resources for your classes. W3Schools (www.w3schools.com) is a fantastic online resource that covers many aspects of web development such as HTML, CSS, JavaScript and also more advanced topics such as SQL, PHP and JQuery. All of the languages have a steady and interactive stream of lessons with working examples for you to review and inspect, line by line. In class this is a great resource for self learning.

The site also provides an excellent series of references for each of the languages, including HTML and HTML 5, highlighting

elements and their compatibility across the many browsers and platforms that exist.

If you're just taking your first steps with web development then there are two essential resources provided by Mozilla. X-Ray Goggles, (<https://goggles.webmaker.org>) is a JavaScript tool that enables anyone to peek at the code that makes up web pages. You can even change the content on the page for use in class – try changing the headlines of a news website, for example.

Another great resource from Mozilla is *Thimble* (<https://thimble.webmaker.org/en-US>) which is an HTML editor in your browser. While not as feature rich as Google Coder, *Thimble* is a step up from using a text editor on a computer and works with most modern web browsers.

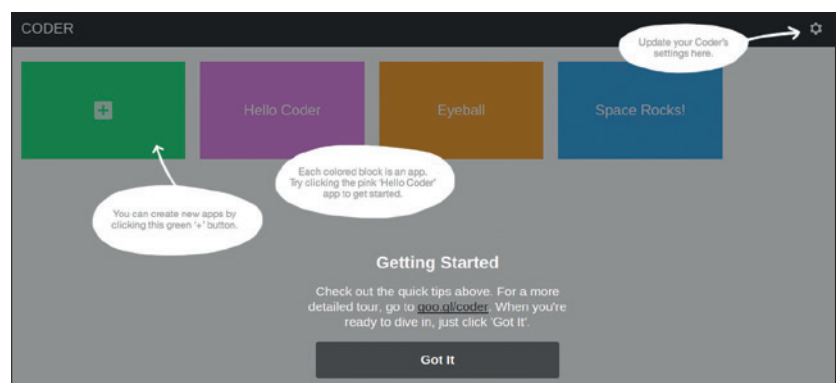
```
</head>
<body>
<h1>Hello World</h1>
<p>Your html goes here.</p>
</body>
</html>
```

So here we have a selection of elements. We start with **<html>**, which instructs the browser that we have written an HTML document. Next we have **<head>**, which performs the tasks that happen behind the scenes, such as loading JavaScript and linking to CSS documents. After the head we have **<body>**, which contains the elements that will be visible in our project. In this case we use **<h1>** to create a large headline that says "Hello World". For all of the tags, we must open them and then remember to close them correctly, for example **<html>** is closed by **</html>**. You may have noticed **<div>** tags dotted around the code. These are tags that divide the HTML page into sections; you can see one called **pagecontent** that contains all of the elements in the document. Later on we'll create our own to contain part of our page.

Now that we understand a little HTML, let's start building our website. We will start by editing the code that is in the body of the HTML document. You can see the **<h1>** tags. Change the contents to:

```
<h1>The Raspberry Pi Computer</h1>
```

Google Coder will display a great navigation tutorial when you first login - take your time to read what it says as it provides lots of useful information.





When you first create an application, it will create a default HTML framework for you to work inside of.

```

CODER Raspberry Pi
1 <!DOCTYPE html>
2 <html>
3 <head></head>
26 <body class="">
27 <div class="pagecontent">
28 <h1>Hello World</h1>
29 <p>Your html goes here.</p>
30 </div>
31 </body>
32 </html>
    
```

On the next line we have `<hr />`. This creates a horizontal line on the page, but this tag is different as it does not have a closing tag, rather it is a self closing tag, denoted by the `/` in the brackets.

Our next line of code is a `<div>` element, and this one has the class (a method to identify it in the document) of `pi`. This new `<div>` will create a section of the page that is separate from the main body of the document. Inside the `<div>` we will create a smaller headline, which asks the question "What is the Raspberry Pi?"

```

<div class="pi">
  <h2>What is the Raspberry Pi?</h2>
    
```

Again we use another `<hr />` tag to create a horizontal line to divide our headline from the main text. Creating paragraph text, as in the main body of text, is achieved using `<p>` tags:

```

<hr />
<p>The Raspberry Pi 2 is a powerful single board computer from the Raspberry Pi foundation.</p>
<p>It comes with</p>
    
```

HTML can display many different styles of data and one of the simplest styles is a list with bullet points. In this project we will use an unordered list to generate bullets, but you could use an ordered list to create a numbered list.

To start a list we first use the `` tag to say that we are creating an unordered list. Then for every item in the list we create a `` tag (list item) that includes the text for that item. Each item in the list will need `` in order to be closed correctly. Lastly we close the list using ``:

```

<ul>
  <li>BCM2836 System On A Chip (SoC) consisting of ARM7 quad core CPU running at 900Mhz per core.</li>
    
```

```

CODER Raspberry Pi
1
2 $(document).ready( function() {
3
4   //This code will run after your page loads
5
6 });
    
```

JavaScript is a great language to learn, and Google Coder enables you to test out your projects.

```

CODER Raspberry Pi
1
2 .pagecontent {
3   padding: 24px;
4 }
    
```

The basic CSS template is sparse but ready for you to edit and make your own.

```

<li>1GB or DDR2 RAM running at 450Mhz</li>
<li>4x USB 2 Ports</li>
<li>1x 10/100 Fast Ethernet</li>
</ul>
    
```

Our next element is a rather lovely picture of the new Raspberry Pi 2, and to display it we need to use the `` tag. Type in:

```


    
```

We now close the `pi` div using `</div>`.

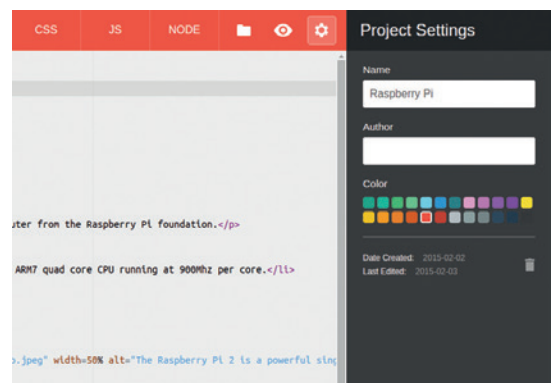
Our last line of HTML is a simple button, which we will make interactive using JavaScript:

```

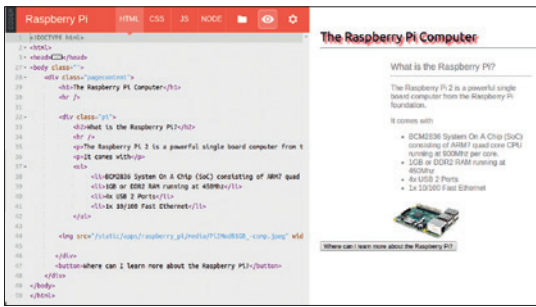
<button>Where can I learn more about the Raspberry Pi?</button>
    
```

Cascading style sheets

We now move on to the CSS tab. CSS is a powerful layout and customisation language that enables a plain HTML page to be transformed into a responsive and beautiful page. CSS is a wonderful tool – it enables even the most basic and plain page to become a stylish experience.



Exporting your application is achieved by clicking on the 'cog' icon from inside your app. From there you can see an arrow in the bottom right, click there to download the project to your computer.



Inside your application you can see a preview of how it will look by clicking on the 'eye' icon – this opens a split-screen preview of your code and the app.

CSS has the following syntax:

```
selector {
  property: value;
}
```

We can select individual elements on a page using the "selector"; we then say what we would like to change about the element and then enter the value.

Here's all the CSS for our page:

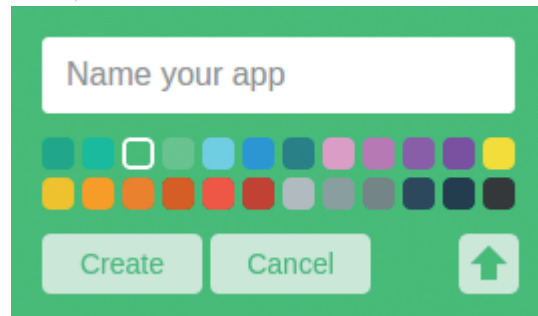
```
.pagecontent {
  padding: 24px;
  background-color: white;
}
h1 {
  color: rgb(0,0,0);
  font-family: 'Ubuntu', sans-serif;
  text-shadow: 5px 5px 5px #ff0000;
}
.pi {
  font-family: arial, verdana;
  color: grey;
  width: 50%;
  margin-left: auto;
  margin-right: auto;
}
```

Our first section of CSS controls the **pagecontent** – it sets the padding of elements and the background colour to white. Colours can be input as names, hex or RGB values. The next section, **h1**, controls the largest headline. We're using an RGB value of 0,0,0, which is black. To make our title look more snazzy we will use a text shadow effect to give it a red glow, which this time is written as a hex value **#ff0000**.

Code for this project

You can find the complete code for this project at our Github repository <https://github.com/lesp/LinuxVoice-Issue-14-GoogleCoder>. Those of you unfamiliar with *Git* can download the complete package as a Zip file from https://github.com/lesp/LinuxVoice-Issue-14-GoogleCoder/blob/master/raspberry_pi.zip

You can easily import the project into Google Coder by clicking on the green new app block in the top-left of the main screen. You will see an up arrow; click there and navigate to the downloaded file. Select the Zip file and the project will be uploaded to your Raspberry Pi and instantly opened ready for you to edit.



To import a project into your Google Coder, click on the green block on the main menu and click on the arrow to open a menu.

Our last section of CSS controls the **pi** div that we created in HTML. To select a div in CSS we must add a full stop before the div name, which constitutes the selector for this element. Our first change is to change the font for this div – we used **font-family** to advise CSS which fonts it should try. If a font is not available then the next font in the list is used. Next we change the text colour to grey. Our last three lines of CSS control the width of the div, in this case 50% of the relative screen size, and to centralise the content we set the margins for left and right to be automatic, giving us a pleasing central column of content.

JavaScript – where the magic happens

Now we come to the JavaScript that controls the button we created in HTML earlier. The button is a method of input, and now we must create an action to happen when it has been pressed. Our JavaScript code to enable the button looks like this:

```
$(document).ready(function() {
  $('#button').click( function() {
    alert("To learn more about the Raspberry Pi, pick up a copy of Linux Voice");
  });
});
```

We start with our first line, which connects our JavaScript code to the HTML document that we wish to work with. We need to do this before we can proceed any further. We now move on to the second line of code, which creates the functionality for our button. We have our button placed on the web page and we instruct the code to look for an event, in this case when the button is clicked. When clicked, the event triggers the next line of code to be executed. We trigger a pop-up dialog box to be displayed on the screen. This is called an alert, as they are generally used to alert the user to an issue, for example alerting the user to an incorrect password.

You will notice that the first two lines of JavaScript mention functions; these are actions that are called when an event occurs, for example the button click, this is given the name "callback".

Congratulations, you have created a simple web page using HTML, styled the content using CSS and added interactivity using JavaScript. 🎉

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

VECTOR GRAPHICS ON THE WEB, FOR THE WEB

MARCO FIORETTI
WHY DO THIS?

- Discover a simple, multiplatform graphic editor that is a great self-training tool.
- Learn a graphic format that is finally ready to take over the web.
- Add images to your website that will look great at any size.

Scalable Vector Graphics are here, and they aren't going away – so learn how they work with an excellent Free Software tool.

Vector graphics are digital images that computers render by executing drawing statements, instead of just copying huge arrays of coloured pixels to the screen. The second method, called “raster” graphics, is the one used by traditional image formats like GIF, JPEG and PNG. The first part of this tutorial explains how vector graphics work and what their advantages are. The second presents a basic, but ubiquitous open source editor to produce and study these graphics.

So what are vector graphics anyway? Have you ever considered how JPEG digital photographs work? Apart from metadata like timestamps and author names, such files just contain an ordered list of all the points (pixels) that compose the picture, each complete with its colour and coordinates. This structure is simple and has one big advantage but even bigger drawbacks. When you use pixels, you can describe any image, from portraits and charts to tropical landscapes, with as much detail as you want. To add more detail, just add more pixels.

Of course, since compression can't do much on images without regular patterns, this greatly increases the file size, which is really bad in this wireless age of often slow connections. Besides, no matter how many pixels a raster image contains, they are never enough to avoid deformations when the image is zoomed, or displayed on screens with different form factors.

Vector graphics completely avoid these problems for all images that are drawings rather than

“photographs”: this includes charts, diagrams, logos, comic-like illustrations and most clip art. Vector graphic files are lists of textual instructions like “let's have a blue five-pointed star on a red background, in the right half of the image, with a height 80% of the total height”.

See the trick? A set of descriptions like that couldn't care less about the size and form factor of the screen: if executed properly, it will always produce a 100% sharp image, everywhere. Want 100 stars instead of one? Just repeat that one command 100 times. And you get all this from a tiny file that only contains plain text, which can be generated by software.

Vector graphics were already great for non-photographic images on the web when they first appeared, more than a decade ago. The arrival of HTML 5, even on mobile terminals, has made them even more interesting. The reason is that HTML 5 pages and applications can directly embed vector graphics inside themselves (it's all text, remember?) and also quickly manipulate them in real time, reacting to user input, with JavaScript.

Your first vector graphics editor: SVG-edit

The only format we need to care about in this tutorial is called SVG, that is Scalable Vector Graphics. While the undisputed king of SVG design with free software is *Inkscape*, here we present another editor, called *SVG-edit* (<https://code.google.com/p/svg-edit/>).

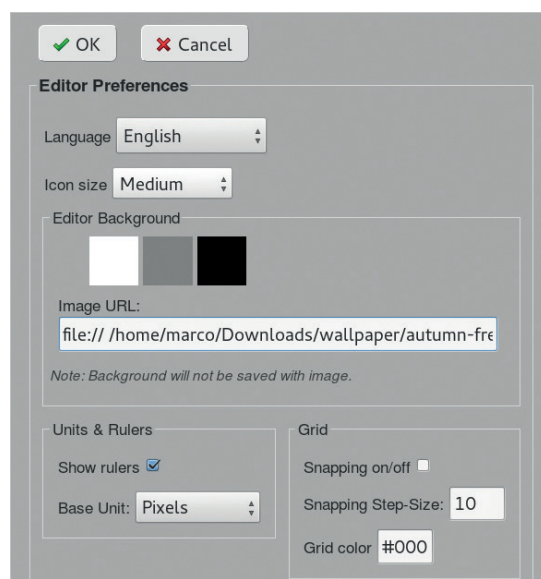
We are doing this because we feel that *SVG-edit* is a better application for beginners to learn about vector graphics, and also flexible and “web-ready” in ways that *Inkscape* cannot match.

SVG-edit has far fewer buttons and menus than *Inkscape*, but still enough to do useful work. You can quickly practice all the essential operations without getting confused by too many options. At the same time, since the user interface has the same general structure as *Inkscape*, it will prepare you to use it.

From a technical point of view, *SVG-edit* is a mix of HTML pages and JavaScript code that runs in any modern browser. There's nothing to install; just load the web page that contains the stable version at <http://svg-edit.googlecode.com/svn/branches/stable/editor/svg-editor.html>.

To begin with, it is very easy to embed *SVG-edit* in your own website with one line of HTML code like this:

```
<iframe src="http://svg-edit.googlecode.com/svn/branches/stable/editor/svg-editor.html" width="750" height="600"/>
```



The first thing to do in *SVG-edit*: study its simple (but important) preferences panel and test it until you find the combination that works for you, because it can greatly impact usability.

You can use *SVG-edit* without internet access. Just download the current version from <https://code.google.com/p/svg-edit/downloads/list>, install it on your local network, and all your relatives, students or colleagues will be able to use it, as long as you also made a local copy of the JavaScript libraries that *SVG-edit* needs, and patched its source code to point to them.

This simple, more or less self-contained architecture also makes it easy to integrate *SVG-edit* in desktop applications. The most popular example is the HTML editor *BlueGriffon*, which uses *SVG-edit* for vector graphics design.

Let's draw with SVG-edit

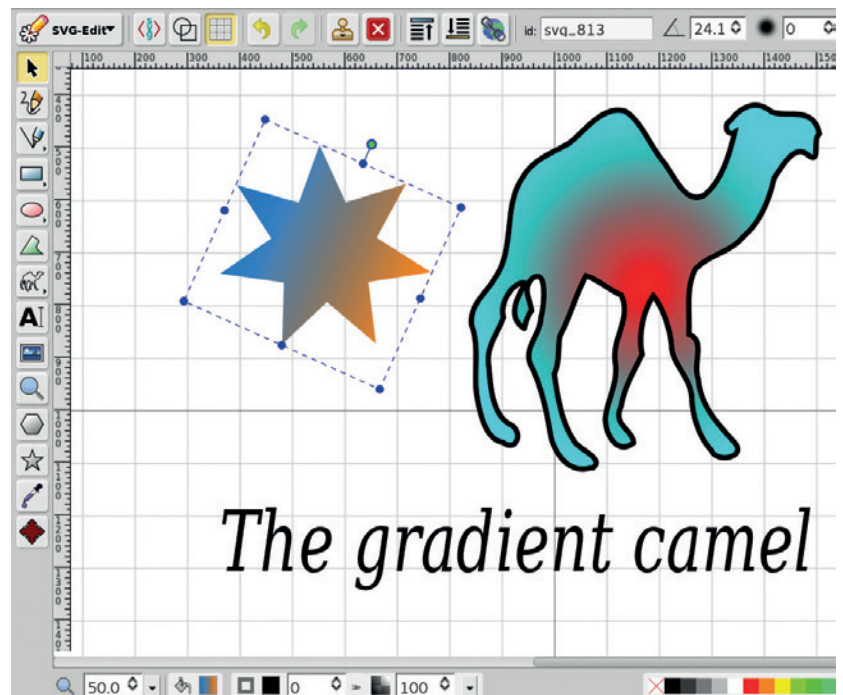
Being an in-browser application, *SVG-edit* is slower than a real desktop application. Keyboard shortcuts for certain functions will be available only if your browser hasn't already mapped them to some of its own functions, and some combinations of size, shape and zoom level of the browser window in which *SVG-edit* runs may make some buttons in the top bar overlap. You may, however, change the size of all the icons in the program configuration panel, accessible from the top-left main menu.

The first time you start this program, it will ask you if you want to store preferences and SVG content on your computer. Accepting is, of course, the only way to not restart from scratch every time.

The user interface of *SVG-edit* is relatively simple. The central drawing area (the canvas) is framed by an edge that hosts one set of functions per side.

At the top there is the main menu, with some "always-on" buttons and a context-sensitive toolbar. Two of the constant buttons are for Undo and Redo; there's one to edit the XML source, and another two to show or hide the drawing grid and the wireframe structure of the graphics.

The bottom area is mainly devoted to colour management. The buttons for all the main tools (cursor, pen, shape library, insertion of text or circles, rectangles and other geometric figures) sit on the left side. Since they are quite intuitive, and equipped with tooltips, we won't spend much time describing them.



Finally, the right-hand side hides the menus for layer management, which we will cover in a moment.

It is possible to load SVG graphics, or include raster images, in the current canvas. Saving your work in either SVG or PNG format is also possible, but works a bit differently than in native desktop applications. When you tell it to save or convert the content of the canvas, *SVG-edit* will open it in a new tab of the browser. You will have to save the content of that tab, as one file with a **.svg** or **.png** extension, by yourself.

Basic operations

In vector graphics, simple geometric objects are defined by their formulas and/or core properties such as radius and centre position for a circle, number of points for a star and so on. To create such objects, click on the corresponding button on the left, then check the drawing parameters that will appear in the top bar. If you don't like their default values (for example if you want a star with 10 points instead of 5), change them as you wish, then drag the cursor on

This is what *SVG-edit* looks like while you work. Notice the context-sensitive top toolbar, the many handles available to move an object and the available gradients.

LV PRO TIP

Plan and configure your layers carefully! The better they match the conceptual structure of the complete graphic, the easier it will be to draw or update it!

The power of scalable vector graphics for the web

Version 1.0 of the SVG standard was published in 2001, but remained largely ignored for years. The reason is simple: there were very few applications that could create such files, and even fewer browser plugins to display them. This situation only started to change in 2008, when *Firefox* and *Safari* gained native SVG support.

Today, SVG isn't yet fully supported as well as a 14-year-old open standard really should be. However, it is already usable for a bunch of very cool and useful applications.

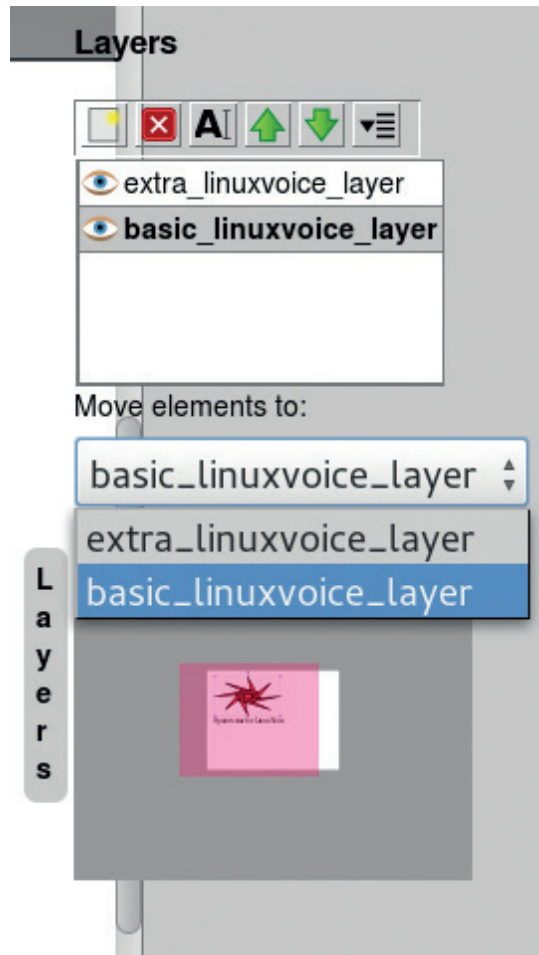
What happens when, thanks to SVG, all the text inside all the graphic elements of a web page is actual plain text, instead of bunches of pixels? Easy: search engines can finally analyse that text too, without errors and with the same accuracy as they already do with the page content. You can

place a long, complex explanation exactly where you want in a graphic, and they will still be recognised as food for their indexes by Google and friends.

At another level, when combined with CSS (Cascading Style Sheets), SVG can give websites backgrounds and decorations that look consistent at any resolution and with any kind of screen, because they can be redrawn on the fly.

The last SVG cool technique we suggest that you learn, after you have mastered the basics, is sprites: single files that contain many independent graphic elements, formatted in a way that a browser can easily extract and use each of them separately. Sprites can make the rendering of complex web pages much faster, and can be used to create simple animations.

Layers are essential in a vector graphic editor, to avoid errors and work faster, without losing track of some object. Remember to name your layers properly!



LV PRO TIP

Path editing is an SVG operation that is a little obscure, but worth practicing. Moving the nodes of a path or linking them to smooth it can be boring, but produces beautiful lines that seem drawn by hand.

the canvas, to define the area where you want them to appear. *SVG-edit* will do the rest.

Depending on the version you use, and on how you configured it, *SVG-edit* may also offer some ready-to-use vector clip art when you click on the library button.

Text? In *SVG-edit* it's just another object. Click on the "A" button, change the default face, size and so on in the top toolbar if necessary, then place the cursor where needed, and type.

Once you have created objects, you can associate hyperlinks to them. You can also move and resize objects as you want just by dragging the whole box that appears around them when they are selected, or any of its corners. The main handle on the top of that box is used to rotate the object. If turning that handle isn't precise enough for you, type the rotation angle you want in the top toolbar. If your goal is relative alignment of objects to one another, use the alignment menu on the right of the top toolbar.

The stamp button is used to clone (only one clone per click, unfortunately) the current object. To move an object from one layer to another, select it and then choose "Move Elements To" in the Layers panel.

There are two ways to do it: the lens on the left enlarges whatever was selected until it fills the whole canvas. The "zoom level" menu at the bottom enables you to zoom in (or out) in more gradual steps.

Please note that we have used the term "object" in the most possible generic sense here. In fact, any

group of objects can be bundled to form a composite object. The button that groups all the selected objects is the one with the two overlapping rectangles.

Layers

Using layers in vector graphics is a must, in at least two very common cases. One includes all the times you end up with so many objects that having them all as one bundle makes it hard to see or select the one you want to edit in any given moment. Defining more layers, and placing only a few objects in each of them makes you work faster, and is also safer, because only the objects in the current active layer are editable.

The other reason to use layers is when you need animated images, or images in which some group of elements cover other groups only partially.

When you drag the "Layers" vertical label on the left, *SVG-edit* opens the Layers Management boxes shown left. As with the rest of the interface, the buttons here are simple: they create, delete, name or reorder layers as you wish. One bit of advice: always use the naming function to give all your layers descriptive names. You and any other future editor of your graphics will be grateful. If you need to change the way in which several objects inside the same layer overlap, just select the one you want to place on top (or bottom) and click on the stacking buttons in the top toolbar.

Paths and path editing

Sometimes there is no combination of straight lines and geometric shapes that will exactly draw the object you want. The solution to this problem is to convert lines to paths, or draw them from scratch.

Paths are combinations of short elementary segments that can be either straight, or sections of regular curves expressed by mathematical functions. Paths can be built and edited in several ways. You can draw freehand and then convert the result to a path, for example. The curved segments of a path may be modified by dragging their constituent nodes.

When you click on the Path tool in *SVG-edit*, the top toolbar will add fields to change the coordinates of

What to read next

The best sources to learn the insides of SVG and understand its full potential are, in our humble opinion, one website and one book. The website is the SVG home page (www.w3.org/Graphics/SVG), which hosts the official specification and other useful material. The book is *SVG Essentials, Second Edition*, by JD Eisenberg & A Bellamy-Royds, published by O'Reilly, 2014.

If you want to extend or customise *SVG-edit*, go to the official wiki (<https://code.google.com/p/svg-edit/wiki>) and read the pages titled "Extension Docs" and "Config Options". The first has a simple, but complete example of how to write an extension. The second lists plenty of options to make the program work just like you want.

Finally, there's one application of SVG for the web that we haven't covered here but that deserves a whole tutorial: embedding interactive, spreadsheet-like charts in web pages. Visit <http://pygal.org> to see what we mean.

each “node”, which is a junction of elementary segments. Another path operator that deserves explanation is the one for “linking control points”: this expression indicates the modification of two segments that connect at the same node (the control point) in such a way that they both have the same tangent in that point. The practical effect of this operation is to make that part of the overall path look smooth, without apices or other discontinuities.

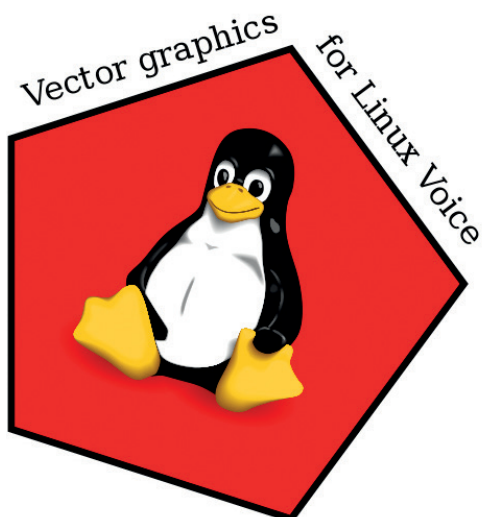
Colours and gradients

Colour assignment in *SVG-edit* works pretty much like in any other basic graphic editor. You can define the fill colour and the border colour of each object from a predefined palette, or by typing a valid HSB or RGB value in the associated input box. To make this change, you must click on the bucket icon first; the eye dropper button changes both the active filling and line (stroke) colours of the current object(s).

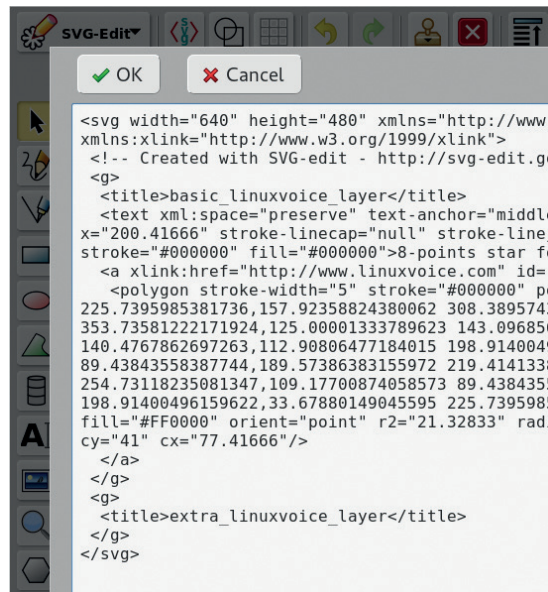
The funkiest colour-related functions of *SVG-edit* are the ones that “mess” with colours: you can fill an object with a pattern varying between two colours, from one extreme to the other (linear gradient) or from centre to borders (radial gradient). The panels to do this open when you click on the bucket icon at the bottom of the screen. The Gaussian Blur input box on the top, instead, applies a user-configurable Gaussian blur to an object, which is Math lingo for “make this object look as if we were seeing it through thick fog”.

Use the source!

The great power of SVG graphics is accessible from the Show Source button of the main *SVG-edit* toolbar: you can see and edit, or just copy and paste in other editors, the whole source code of the current drawing. We strongly suggest that you use this feature to look inside every SVG graphic you draw by yourself, or find online. Even if you are 200% sure that you will never want or need to edit SVG sources by hand in your whole life, looking at that code (which is verbose, but much easier to understand than you may think) will



Object groups, crisp lines, rotated text... here's an example of some of the functions of *SVG-edit*.



Under the hood, all SVG graphics are plain text like this, that you can easily study or tweak with any editor – check for yourself.

teach you more about vector graphics than thousands of mouse clicks.

Is it possible to extend the functionality of *SVG-edit*? Yes, of course. You can add buttons made of SVG graphics to the left or top toolbars of *SVG-edit*, and bind them to generic JavaScript code, saved in the *SVG-edit* sources, or (much better) in a separate file.

Advanced configuration

An extension saved in a single JavaScript file, placed in the **extensions** subfolder of the *SVG-edit* installation, will be loaded if you add the name of that file to the URL of the editor, as in this example:

```
http://example.com/my-svg-edit/svg-editor.html?extensions=my-extension.js
```

Calling *SVG-edit* in this way is extremely simple. However, it disables all the other extensions that the program may have otherwise loaded. Depending on your needs, this may be a bonus or something to avoid at all costs. Luckily, there is a way to load only the extensions you want out of those placed in the **extensions** subfolder of *SVG-edit*: list them in the array of the same name that is defined in the source file called **svgedit.compiled.js**.

Other options, like the default size and background colour of the canvas, can be set in two ways. You can add all of them to the URL:

```
http://example.com/my-svg-edit/svg-editor.html?dimensions=800,600&&initFill[color]=FF0000
```

or write them in a file called **config.js**, in the root folder of *SVG-edit*, inside a JavaScript hash called **setConfig**:

```
svgEditor.setConfig({
  dimensions: [800, 600],
  initFill: {
    color: 'FF0000'
  }
});
```

Marco Fioretti is a Free Software and open data campaigner who has evangelised FOSS all over the world.

LV PRO TIP

Never forget to have a look at the SVG source code of every graphic you create. With tools like *SVG-edit*, or any decent text editor, it is very easy, and worth the time you spend on it.

HOW YOUR COMPUTER WORKS: INSIDE AN X86 CHIP

You press a switch, and Linux starts booting. What's going on under the hood in the meantime? Let us explain.

WHY DO THIS?

- Understand why Linux works the way it does
- Troubleshoot occasional problems better
- Earn yourself some geek points

Linux has a long-established reputation of being the operating system for hackers. It became unnecessary to know what happens behind the curtains of major desktop distributions a while ago, but it is still beneficial to understand how computers really work. Being able to decipher cryptic error messages means you can diagnose Linux problems (let's face it – this happens from time to time) much more quickly. It is also fun and not really difficult if you have some programming background.

We often hear that Linux is ideal for education, as it doesn't attempt to hide the inner workings from anyone curious enough to look at them. Let's take this one step further and learn something from our favourite OS. So, brew yourself a cup (or glass) of whatever you prefer, and let's get started.

A bird's eye view

Although your computer is undoubtedly a very sophisticated device, the operations it performs are (in essence) quite simple. The Central Processing Unit, or CPU, continuously samples program instructions from memory and executes them on data that also comes from RAM. Most modern computers (and even smartphones) have multicore CPUs, which are essentially multiple processors on the same die

sharing some resources, like caches. From a programmer's viewpoint, multicore and multiprocessor don't differ too much, and in this tutorial, we'll use both terms interchangeably. Processor instructions are rather low-level: you can load and store bytes of memory, do basic maths, jump unconditionally (like **goto**) or conditionally (like **if-else**), and maybe calculate CRC32 checksums or do AES encryption, if your CPU provides these extensions. However, there are no higher-level functions, like "convert integer to string". Some CPUs (like x86) can access memory directly, while others (ARM) operate on registers.

Registers also store the processor's state, and carefully saving and restoring them is how Linux switches tasks (threads or processes).

An x86 CPU can operate in different modes. Sometimes, programs are granted access to the whole memory, and no address translation is performed. This is known as 'real' mode, and is used during the boot phase or in older operating systems like MS-DOS. Other times, the CPU may check memory access rights and prevent one program from touching another. This is called 'protected mode', and it's often combined with paging, or address translation, to produce the paged protected mode that Linux (and other major OSes) run in.

Finally, most modern CPUs are 64-bit, but can run in either 32-bit 'compatibility' or 64-bit mode (the 'long mode' term refers to both). The memory address and register size are determined by whether the processor is 32- or 64-bit; 64-bit CPUs can address more memory and handle data in larger chunks, which usually means better performance.

Add-ons

Computers also have peripheral devices, like video cards or network adapters. Nowadays, they are often built-in and not separate extension cards, but for our discussion, it doesn't really matter. To communicate with these devices, programs running on the CPU use I/O ports or memory-mapped registers (MMIO). MMIO is memory range that is accessed like the rest of RAM, but resides on the device rather than in a DRAM bank. If a device needs CPU attention, it sends an interrupt, which can be as simple as putting a selected wire voltage to low or high, or as sophisticated as a special type of message within the PCI bus. Either way, this signal ends at the local APIC, or Advanced



Even the high-end x86 computer you may have on the table today follows the architecture John von Neumann described back in 1945.

Provided by LANL, public domain (see <http://commons.wikimedia.org/wiki/File:JohnvonNeumann-LosAlamos.gif>)

Programmable Interrupt Controller. It's integrated with the CPU (or its core) and uses clever algorithms to decide when to tear the CPU away from what it's currently doing and ask it to service the interrupt. APIC can also serve as a source of interrupts itself: it has a built-in timer and facilitates sending inter-processor interrupts or IPIs.

More on registers

A register is essentially a small (typically 64- or 128-bit) and fast memory cell built into the CPU. Some registers have a predefined meaning or purpose, while the others can be used to store arbitrary data. The latter ones are usually called general-purpose registers, or GPRs. The number of registers available and their names (also known as a register file) are defined by the CPU architecture. A 64-bit x86 CPU has 16 general-purpose registers: RAX-RDX, RDI, RSI, RBP, RSP, and R8-R15. If you're wondering about the names, then historically the Intel 8086 processor had 16-bit AX, BX, CX and DX registers. Intel 80386 (I still have this one in the attic!) introduced 32-bit mode. Registers also became 32-bit wide and got an E-prefix ("E" stands for "Extended"). The R-prefix and numeric registers were introduced with 64-bit mode, to increase size of the register file. In 64-bit mode, E-prefixed registers become lower halves of their R-prefixed counterpart.

Special-purpose registers come in several flavours. First there's RIP, or the program counter, which stores the memory address immediately after the current instruction. The instruction address is defined relative to the code segment base, which is available via the CS register. There are other segment registers, like DS for data, SS for stack, or GS, useful when switching from user space to kernel space in 64-bit mode. Segments date back to 16-bit times where a single word-sized register wasn't able to address memory beyond 64k.

Nowadays, with registers being at least 32-bit wide, segments are not that important (for more on this, see Mike's assembler tutorial on page 106). In fact, they are mostly ignored in 64-bit long mode. However, some bits of the CS register are still recognised; these include privilege level field (so segments can be used as a memory access control mechanism), and the 'L' flag, which is set for segments containing 64-bit code.

Registers in the wild

CPU registers are a scarce resource. As they are much faster to access than memory, using them for data operations drastically increases performance. However, data begins and ends in memory, and the associated overhead can make a whole game not worth the candle. That's why compilers use clever algorithms to allocate registers for a program's variables in the most optimal way. The C programming language even provides the `register` keyword, which can be used as a hint for the compiler. It was useful back in older days, but now most optimising compilers are smart enough and simply ignore it.



Most modern CPUs are in fact multiprocessors. This AMD A10 (Kaveri) chip is no exception: it has four CPU and eight GPU cores.
Photo: Yulia Sinitsyna

This way, legacy 32-bit processes can run in a 64-bit OS if they are assigned a CS with L=0. Library code, however, shares the CS register with the process it is linked to, and as plugins are often implemented as shared libraries, there's no way to run a 32-bit plugin in a 64-bit host (except by putting it inside a separate 32-bit process).

Then there are the control registers – CRx. Officially, there are sixteen of them, but only few are currently used. CR0 determines whether the CPU is in real, protected, or paged mode, or a combination thereof. The CR3 register contains a pointer to the page table root used to translate addresses in paged mode. We'll cover this and the CR2 register shortly.

When the CPU is in protected mode, only privileged code (the operating system kernel) is allowed to change control registers. This way, for example, Linux processes' virtual address spaces are kept isolated from each other.

Finally, there are model-specific registers, or MSRs. As the name suggests, different CPU models may have different MSRs even within the same (x86) architecture. These registers are widely employed to support advanced features that weren't initially part of the x86 architecture. This includes 64-bit mode, or the x2APIC interrupt handling found on modern Intel CPUs and vital for low-latency virtual machines. Model-specific registers are referred by their numbers and are 64-bit wide.

Back into labs

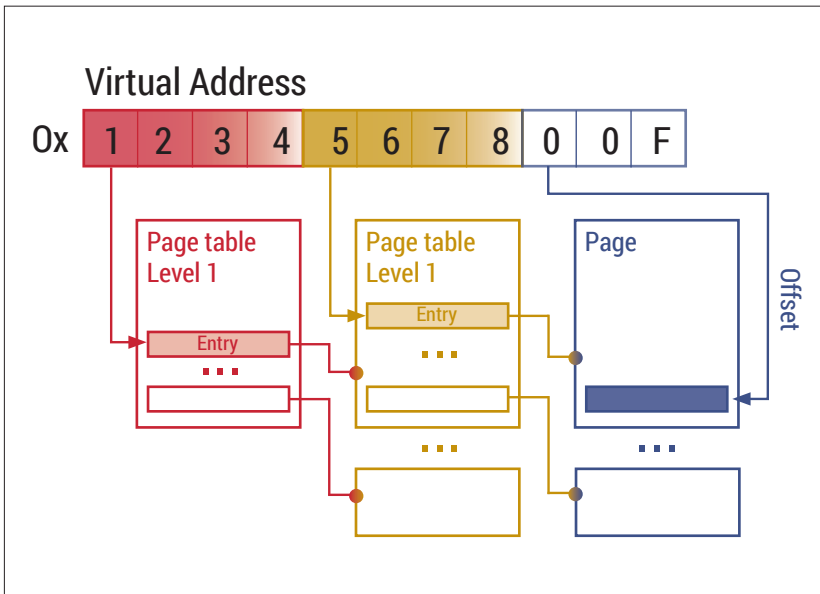
There's much that can be said about registers, but it's time to practice. Let's start with a simple experiment. Open a terminal and type:

```
$ cat /proc/cpuinfo
```

You'll get a lot of information about your CPUs, somewhat like this:

```
processor      : 0
vendor_id    : GenuineIntel
...
flags        : fpu vme ... vmx ... nx ...
```

Many of these bits of data are obtained with CPUID



An example of address translation using two-level page tables. Offset within the page is not translated but added to the result. copyrighted by Yulia Sinitsyn

processor instruction. It queries CPU functions (also known as “leafs”) by number (passed in the EAX register) and receives results in EAX, EBX, ECX and EDX. For instance, the **GenuineIntel** string is returned by function 0. For an AMD processor, like the one in the photograph, it would be “AuthenticAMD”. Bit 5 set in ECX for function 1 means the CPU supports VMX (Intel’s virtualisation technology). Checking that **/proc/cpuid** contains the **vmx** flag is common advice

when *KVM* and *VirtualBox* refuse to start, and now you know where it really comes from. **/proc/cpuid** shows only a subset of CPUID leaves. If you want them all, consider the **cpuid** tool:

```
$ sudo cpuid 0 # 0 is processor number
Leaf   Subleaf  EAX      EBX      ECX      EDX
00000000 00000000: 0000000d .... 68747541 Auth 444d4163
cAMD 69746e65 enti
00000001 00000000: 00630f01 ..c. 00040800 .... 3e98320b
.>. 178bfbff ....
```

Note how “AuthenticAMD” is returned.

You can play with MSRs in a similar fashion, although I wouldn’t recommend writing to arbitrary registers in a production environment because you can hang your machine quite easily. Reading is safer, but unless you know the register numbers it’s very much an “arbitrary value in – arbitrary value out” experience. Consider the following:

```
$ sudo rdmsr 0xc0000080
d01
```

RDMSR is short for “read MSR”; it’s an assembly instruction that the tool was named after. MSR number 0xc0000080 is the Extended-Feature-Enable (EFER) register, and it has various uses. For instance, to enable SVM (AMD’s virtualisation technology), you set bit 12 in EFER to 1. If you are on AMD, start any KVM guest and run the command again:

```
$ sudo rdmsr 0xc0000080
1d01
```

0x1d01 is 1110100000001b, so bit 12 is set. Trying to clear it with the **wrmsr** tool while the guest is running triggers a kernel bug in KVM (don’t say I enticed you to try this):

```
$ sudo wrmsr 0xc0000080 0xd01
$ dmesg | tail
[25360.493222] -----[ cut here ]-----
[25360.493279] kernel BUG at arch/x86/kvm/x86.c:290!
[25360.493323] invalid opcode: 0000 [#1] PREEMPT SMP
```

EFER also enables long mode. Two bits, LME (number 8) and LMA (number 10), indicate that long mode is enabled (E) and active (A). An OS that wants to run in long mode sets LME bit, and the mode is activated as soon as paging is enabled in CR0. So, both LME and LMA are running in a 64-bit system. The value above has both bits set, as well as bits 0 and 11. The former enables fast system calls, and bit 11 is the famous NX (or Non-eXecutable) flag that makes it harder to exploit common vulnerabilities. It’s optional, but we saw that our CPU supports it in **/proc/cpuinfo**.

A word of memory

How a CPU sees memory depends on processor mode. Here, we’ll speak of paged protected mode, as it is the main one Linux runs in. In this mode, memory can be seen as a collection of pages. Put simply, a page is a 4k (or more, like 2MB or even 1GB) chunk of bytes that share common properties, like cacheability or access rights. Many low-level structures, like virtual machine control blocks, start at a page boundary, or are page-aligned. Linux keeps track of every page frame in the system with a special structure (**struct page**): it may sound wasteful, but in reality this uses only a tiny fraction of available memory. Linux also combines pages into “zones”, and you can easily see how they are organised on your system in **/proc/zoneinfo**.

When in paged mode, the CPU also uses special tables to translate the addresses it operates on (or “virtual”) into real ones (or “physical”). A part of the CPU called the Memory Management Unit (MMU) is responsible for this, and the IOMMU does the same for external peripherals. Translation enables each process to have its own private address space, and a kernel to protect itself from user-mode programs.

Prepare your tools

cpuid, **rdmsr** and **wrmsr** are parts of the Intel-developed **msrtools** package. You are unlikely to get them with package manager, but they are small and trivial to compile yourself. Just download the tarball (only 7k!) from <https://01.org/msr-tools>, unpack and run **make**. If you get errors when running tools, ensure you have the **cpuid** and **msr** stock kernel modules loaded, and device nodes created; the **MAKEDEV-cpuid-msr** script bundled with the sources can help with the latter. Root privileges are required, so be careful.

To support this protection, page tables contain access control flags. A page can be marked as present, writable, executable, or available for the kernel only. If code accesses a page that isn't present or is otherwise unavailable, the faulty address is stored in the CR2 register and a hardware exception is thrown. If this happens in userspace, the kernel checks if the page accessed was swapped out to the disk and either reads it back or sends a SIGSEGV to the program (that's the famous "Segmentation fault" error). If the exception occurs in kernel mode, it is considered a serious bug, and the kernel panics.

Translation tables

Translation tables are essentially chained arrays (see the diagram). First, the virtual address is split into several parts (four, if in 64-bit mode). Each part is then used as an index into the table that contains either an address for the next page table, or (at the lowest level) a physical address of the page itself. The exact format of the page table entry differs depending on page size or extensions, like Physical Address Extension (PAE). If PAE is enabled, more bits are allocated for the page's physical address, allowing 32-bit code to reference memory whose physical address is beyond 4GB. This doesn't magically make 32-bit address space more than 4GB in size, but it helped 32-bit servers to have more RAM until 64-bit servers become mainstream.

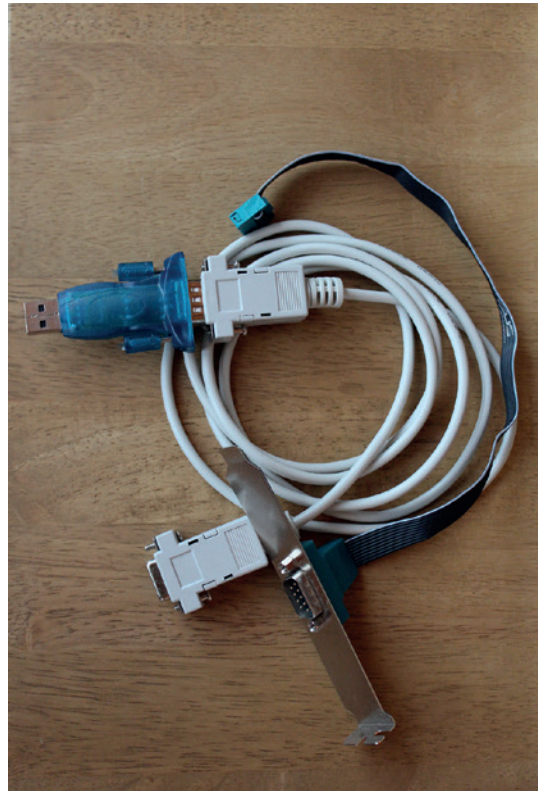
Now let's check which memory regions are defined on your computer (this depends on BIOS, RAM size, and the kernel version):

```
$ cat /proc/iomem
...
00100000-bf680fff : System RAM
01000000-015427b9 : Kernel code
015427ba-018e037f : Kernel data
01a05000-01b2afff : Kernel bss
...
c0000000-feafffff : PCI Bus 0000:00
...
d7100000-d80fffff : PCI Bus 0000:06
d7100000-d710ffff : 0000:06:00.0
d7100000-d710ffff : ath9k
```

Addresses shown here are physical. You can see where the kernel lives, and also the memory range assigned to Atheros network adapter (**ath9k**). If a user space program can modify translation tables and map that address, it could re-program a card. For example, it could send network packets bypassing a system firewall. That's why managing page tables is the kernel's job.

A bit of device

Finally, let's briefly cover external devices. x86 architecture provides what's called I/O space – 64k of addresses (known as "ports") accessible with IN and OUT assembly instructions. They are already enough to send data over serial line. That's why system programmers often use serial ports for low-level debugging – USB, for instance, is much more



Serial ports are easy to program, but with modern PCs, you may need a bit of cabling to actually use them.

Photo: Yulia Sinitsyna.

elaborate. One way Linux lets you see how I/O ports are used should feel familiar:

```
$ cat /proc/ioports
0000-0cf7 : PCI Bus 0000:00
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-0060 : keyboard
0064-0064 : keyboard
...
```

Here, we see that ports 0x60 and 0x64 are assigned to a PS/2 keyboard controller. Surprisingly, we can also use it to reboot the computer! This explains why access to the I/O port from Linux userspace is usually prohibited. However, the **ioperm()** and **ioport()** system calls can make a desired I/O port range (or a whole I/O address space) accessible. Root privileges are generally required, and it is rarely a good idea to use these unless you run some old X server.

Now you have a some understanding of how the computer you use everyday works internally. That wasn't that hard, was it? This tutorial is a kind of high-level overview, but you can find all details in Intel's SDM (Software Developer's Manual) or AMD's APM (Architecture Programmer's Manual), available freely (as in beer) from the respective websites. Linux opens many possibilities to try new stuff you learned. If you come up with some clever experiment, don't forget to share it with us! 📧

Dr Valentine Sinitsyn prefers programming bare-metal but occasionally writes some Python. He contributes to the Jailhouse hypervisor and teaches physics.

LV PRO TIP

If you liked this tutorial, consider getting yourself two books: *Inside the Machine* by Jon Stokes and *Linux Kernel Development, 3rd Edition* by Robert Love.

SHARES WITH SHARED CODE – PART 2

ANDREW CONWAY
WHY DO THIS?

- Work with real-world data
- Prove that you're smarter than a City whizz-kid
- Laugh at fund managers and their obscene money-for-nothing charges

Mix real-world data with our own custom algorithms to make enough money to buy that volcano you've always wanted.

Settle down everyone – we're going to write code to buy and sell shares from a portfolio to ensure that we don't lose money and, hopefully, that we make some money. For simplicity, we'll begin by looking at a portfolio of shares that's identical to those in the FTSE 100, and so the only choices are when to buy or sell shares.

The FTSE 100 is a stockmarket index that is constructed by taking a weighted average of the value of shares for the 100 leading companies registered on the London Stock Exchange. At the start of 1984, when the FTSE 100 began, it was given the value 1000 and by the end of 2014, due to the changes in share values, the FTSE 100 had risen to about 6,500. So if we bought shares in 1984, in proportion to how the FTSE 100 is weighted, and kept our portfolio mirroring the composition of companies represented in the FTSE 100, then if we sold them 30 years later in 2014, we should see a return of about 650% (or a factor of 6.5, if you prefer) on our initial investment. Once you correct this for inflation – the fact that prices have risen over the 30 years – it is still a respectable 300%.

But you'd not get such an impressive return for all periods of the FTSE 100's history. In fact, if you bought at the 2000 peak, which coincidentally was also about 6500, then selling in 2014 would just return the money you put in, and after accounting for inflation you'd be worse off.

Next, let's consider how a simple sell high and buy low scheme might improve things. If we bought £1,000 of FTSE 100 shares in 1984, then sold at the year 2000 peak to receive £6,500, and invested this all again in 2003 when the FTSE 100 was worth only 3,500, then

we'd get $(6,500/3,500)*£6,500=£12,071$ in 2014, ie a return of 1,200%.

Clearly, it is possible to devise an algorithm that can profit from the peaks and troughs of share prices. In fact, we can improve things even more if we sell at the 2007 peak and buy in the trough of 2008/9. But, there's a snag, which might well have occurred to you. It's easy to see peaks and troughs in a graph of historical data, but there's no way to know whether you have reached a peak (or trough) at the time

unless, of course, you can predict the future. That said, it is still possible to devise sell-high, buy-low algorithms that do not rely on knowing you're at a peak or in a trough.

Aim high, start low

Let's now construct our first, very simple algorithm to decide when to buy and sell shares for a single company. It makes one of two decisions: either to spend all available money buying shares, or to sell shares we currently possess. This is of course a terrible strategy, and similar to what a desperate drunk might employ in the wee hours at a casino.

The code, written in Java, is:

```
public class SingleInvestment {
    double money, sellThreshold=2, buyThreshold=-0.05;
    TimeSeries timeSeries

    void invest(double investment) {
        money=investment-timeSeries.purchaseShares(investment);
        do {
            sellShares();
            buyShares();
        } while (timeSeries.next());
        double rawProfit = money+timeSeries.getFinalValue() -
investment;
    }
}
```

The **money** variable will record what's not invested in shares, and the **Threshold** variables are parameters, of which, more later. The **TimeSeries** class handles all aspects of the time series, including iterating through time and purchasing and selling shares, and is designed to keep our investment code clean and readable. In fact, we needn't concern ourselves with how the **TimeSeries** class is implemented.

The **invest()** method takes the amount to be invested and on the very first line splurges it to purchase as many shares as possible with the call to **purchaseShares()**. Only a whole number of shares can be bought, so it returns the actual amount spent, which we use to update the **money** variable. Next, we start the loop which calls **sellShares()** then **buyShares()** repeatedly until **timeSeries.next()** returns **false**, telling us we've reached the end of the data. The **sellShares()** and **buyShares()** methods are as follows:

```
void sellShares() {
```

“It's possible to devise an algorithm that can profit from the peaks and troughs of share prices.”

```

if (timeSeries.getPrice() > sellThreshold * timeSeries.
getPriceAtLastPurchase()) {
    money += timeSeries.sellShares(timeSeries.
getSharesHeld());
}
}

void buyShares() {
    double delta = timeSeries.getDelta(5);
    if (timeSeries.getSharesHeld()==0 && delta < buyThreshold
* timeSeries.getPrice()) {
        money -=timeSeries.purchaseShares(money);
    }
}

```

The `sellShares()` method checks the current price of the shares we're holding, and if the price has risen to more than `sellThreshold` times the price we paid for them, then it sells all shares by calling `timeSeries.sellShares()` returns the amount from the sale so we can add it to the money variable.

The `buyShares()` method is similar, except its condition for buying is twofold.

First, we currently hold no shares, and secondly that delta – the difference between the price now and the price five time-steps ago – is less than `buyThreshold` times the current share price. Applied to monthly FTSE 100 index data from its beginning to January 2015, this algorithm generates £10,982 on an investment of £1,000, ie a profit of £9,982. Since £1,000 in 1984 equates to about £3,000 today when adjusted for inflation, this is a very respectable return, and not too far behind the return achieved with prescient knowledge of peaks and troughs.

But there's still a snag with this scheme: how should the parameter values be decided, and how sensitive is the profit to their values? The answer to the latter question is quite sensitive: if we change `sellThreshold` to 1.5, then we no longer make a profit, but a loss of £2,105, and if we change it to 2.5, we make a below-inflation profit of £379. The values of 2 and -0.05 came from trial and error, guided by a thermally efficient, wet neural network, ie my brain. Also, remember that these values may work well on the history of the FTSE 100, but there's no guarantee that these values will work on future data.

Multiple eggs – one basket

Now let's move up a gear and work with a number of different shares rather than having a portfolio which is mirroring the FTSE 100. For simplicity, we'll restrict ourselves to owning only one of these shares at a time: BPL, ITV.L, LLOY.L, MKS.L, NG.L and TSCO.L. If you want to know more about these companies, you can search for them on yahoo.finance.com or www.londonstockexchange.com, but leave off the .L suffix if you use the latter, because it just means the shares are listed on the London Stock Exchange.

The code used is not too different from before:

```

private static int NO_MORE_DATA = 0, NO_NEW_DATA = 1,
NEW_DATA = 2;

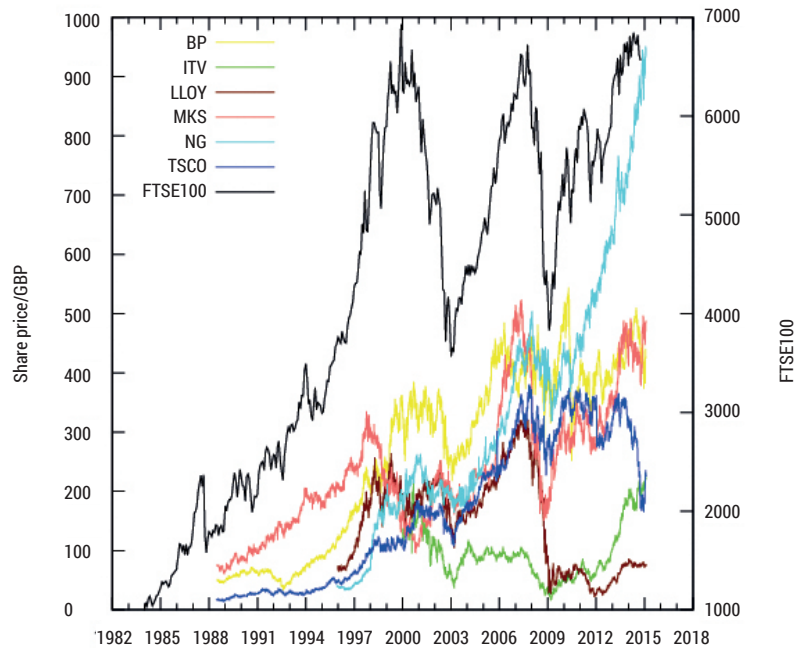
```

How our companies have been doing

Here's a plot of prices for all shares we're interested in, and the value of the FTSE100, which uses a different scale to the right. You can download the data from Yahoo finance. For example, on the command line you can fetch all available data for the share BPL

with:
`wget "http://ichart.finance.yahoo.com/table.csv?s=BPL" -O BPL.csv`

Or you could put that same URL into your web browser. We'll be working with the `Adj.Close` column of the data.



```

ArrayList<StockTimeSeries> seriesList = new ArrayList();
ArrayList<StockTimeSeries> availableList = new ArrayList();
...
public void invest(double investment) {
    money = investment;
    Calendar cal = Calendar.getInstance();
    cal.setTime(seriesList.get(0).getFirstDate());
    int index=-1;
    status = updateAvailableList(cal.getTime());
    do {
        if (index == -1) {
            buyShares();
        } else if (sellShares()) {
            index = -1;
        }
    }
    do {
        cal.add(Calendar.DAY_OF_MONTH, 1);
        status = updateAvailableList(cal.getTime());
    } while (status == NO_NEW_DATA);
    } while (status != NO_MORE_DATA);
}

```

The main difference is that we are not just looping through one time series, but looking at a number of series that are in a List called `seriesList`. The code that prepares this isn't shown here, but it ensures that the series with `index 0` has the earliest start date, and this is the date that we put into the `Calendar` object. We start with `index=-1` which means that we currently hold no shares. Next we call `updateAvailableList()` with the starting date, and it will put all time series

objects with data for that day into the `availableList`, and return a status of `NEW_DATA`, which is one of three constants defined in the class.

The loop then starts, and since `index` is -1 (we have no shares) it attempts to buy some. In later iterations,

after we own some shares, an attempt is made to sell shares instead. Then we start another loop that will repeatedly increment the date by one day and

“This scheme gives a return of £12,140 on an investment of £1,000 – a profit of £11,140.”

then call `updateAvailableList()` until it returns a status of `NO_NEW_DATA`. This winds us past weekends and bank holidays on which the stock exchange is closed and the `availableList` is empty. The loop checks to see if status has not been set to `NO_MORE_DATA`; if it has we've reached the end of all the data and we're done.

The `sellShares()` method is nearly identical to before, except that it now returns `true` if it sells shares, but `false` otherwise. The `buyShares()` method is a little bit more involved than before:

```
void buyShares() {
    for (int i = 0; i < availableList.size(); i++) {
        StockTimeSeries timeSeries = availableList.get(i);
        double delta = timeSeries.getDelta(5);
        if (delta < buyThreshold * timeSeries.getPrice()) {
            money -= timeSeries.purchaseShares(money);
            index = i;
            return;
        }
    }
}
```

It loops through all series in the `availableList` and looks for a drop in share price of sufficient size in the same way as before. As soon as it finds one share with such a drop, it will purchase as many shares as it can and return.

The output from using this class, investing £1,000 initially, with `sellThreshold=1.5` and

`buyThreshold=-0.05`, is:

```
1988-08-16,1000,0,3 BUY: BPL.csv
1990-08-17,-,1522,3 SELL: BPL.csv
1990-08-21,1522,0,3 BUY: MKS.L.csv
1992-04-13,-,2315,3 SELL: MKS.L.csv
1992-04-30,2315,0,3 BUY: TSCO.L.csv
1995-07-07,-,3507,3 SELL: TSCO.L.csv
1995-09-22,3507,0,3 BUY: TSCO.L.csv
1997-05-16,-,5274,5 SELL: TSCO.L.csv
1997-05-20,5274,0,5 BUY: MKS.L.csv
2006-03-10,-,8051,6 SELL: MKS.L.csv
2006-03-31,8051,0,6 BUY: ITV.L.csv
2013-07-11,-,12150,6 SELL: ITV.L.csv
2013-08-15,12150,0,6 BUY: MKS.L.csv
Final money=0
Final value of held shares=13628
Raw profit=12628
```

Each line of the output records a transaction: the date of transaction, value of shares held, money held (after this transaction), size of the `availableList`, and a short text description. You can see here that this scheme is our most successful yet, bringing a greater than 12-fold return, before inflation.

A proper portfolio investment

Let's now implement a true portfolio of shares where we can hold shares of several companies at once. We will decide when to buy and sell shares exactly as above, but a new decision has to be made: how many shares should we buy or sell? Adhering to the KISS principle (Keep It Simple Stupid!), let's plump for creating two new parameters: `sellFraction` and `buyFraction`. The first means that when we decide to sell a particular share, we sell the number of those shares held times `sellFraction`. Similarly, for `buyFraction`, once a decision is made to buy shares, the amount of money to be spent is set equal to `buyFraction` times the amount of money we currently hold.

The `invest()` method hardly changes, and in fact ends up becoming simpler because we do away with the `index` variable, and `if-else` statements testing it inside the main `do-while` loop are replaced with:

```
buyShares()
sellShares()

The buyShares() code hardly changes: we only need to remove the index=i line and in the line above replace money with buyFraction*money. The sellShares() method needs to change a bit so that, like in buyShares(), it loops through all available shares, and takes notice of the new parameter, sellFraction:

void sellShares() {
    for (int i = 0; i < availableList.size(); i++) {
        StockTimeSeries timeSeries = availableList.get(i);
        if (timeSeries.hasShares() && timeSeries.getPrice() > sellThreshold * timeSeries.getPriceAtLastPurchase()) {
            money += timeSeries.sellShares(sellFraction);
            return
        }
    }
}
```

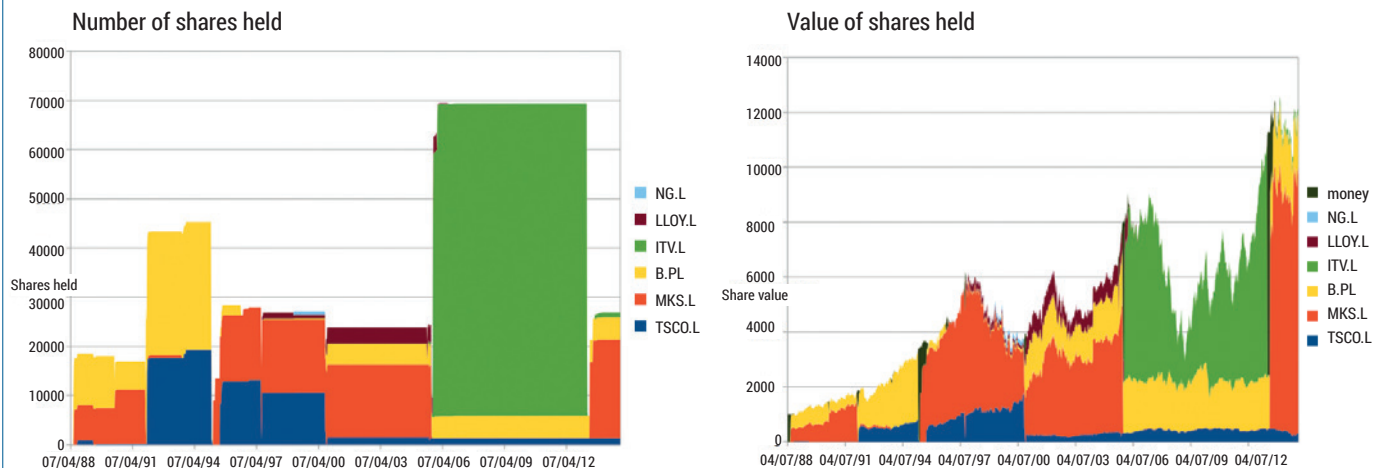
Dark arts of parameters

In building models, whether they are to predict time series, or model the climate, or plan a space mission, it's next to impossible to avoid using parameters. Some parameters, such a physical constant, like one that describes the strength of gravity, can be measured objectively, but others, like the sell and buy thresholds we've used, need to be set empirically – that is by looking at real data. And this causes a problem: we want the parameters to work not just on the data we have, but on any data we throw at the model. This issue extends beyond numerical modelling. For example, it crops up in learning a foreign language. At school my French teacher taught us *“J'ai treize ans et j'habite a Edimbourg”*. If I learned that parrot-fashion, without understanding, it'd soon be useless to me as I wouldn't be 13

and living in Edinburgh for my entire life (as it happens I was 12 and lived in Glasgow!).

The point is that we must not choose parameters so that they only work well on one set of data. In fact, it's always possible to keep adding parameters to a model so it can describe a given set of data perfectly, but when it does, it will almost certainly fail when given fresh data. There are various methods available to set parameters to avoid this pitfall, but they mostly boil down to a simple idea: reserve one set of data for choosing the parameters – the training set – and another set of data for checking that the model generalises well – the test set. If you think about it, this is exactly how school learning proceeds – you are taught on one set of examples, but will be tested on an unknown set of examples in the exam.

How our portfolio evolved



The graphs are stacked, which means that the top of each column is the total number of shares held, with the height of the colour rectangles showing the proportion of each share held. For example, from 2007 to 2013, no shares were bought or sold and the number of shares remained constant at just below 70,000. Most of those shares were in ITV.L, with some in BPL and TSCO.L.

Of more relevance to most people will be the amount of money that our shares are worth. The value of shares held takes on a spiky appearance, reflecting the volatility of share prices. The amount of cash money held is shown too, though it is almost always small in proportion to the total value of shares held.

```
}
}
```

Notice that, keeping to our KISS principle, as soon as we find shares that meet the criterion, we buy or sell then return; only one buy and sell transaction can take place at each time-step.

Running this scheme, with **sellThreshold=1.5**, **buyThreshold=-0.05**, **buyFraction=0.5** and **sellFraction=0.5**, gives a return of £12,140 on an investment of £1,000, ie a profit of £11,141, about £1,000 less than the previous scheme. Although this return is lower, our investment is safer in that with the previous scheme we could have lost everything if the single company we'd invested in went bust.

The graphs of this scheme show some odd features. The most striking is that there were no transactions at all between March 2007 and June 2013. During most of this time, the UK, along with many other countries, was either in recession or enduring a feeble recovery. At first it's surprising that the algorithm didn't buy shares during the start of the recession in 2008, because that was a time when prices were falling. However, on closer inspection you can see that our money was already fully invested when the recession started, so we rode out those years with an unchanging portfolio. It wasn't until 2013 that any of the shares rose in price enough to trigger a sale. This is great example of why investing in shares is regarded as a long term investment, and not a get-rich-quick scheme.


Bank share prices suffered the most pronounced fall in the 2008 recession, so it's particularly pertinent to see how those of LLOY.L – the Lloyds bank - fared at this time. They were bought and sold by our scheme up until 2007, but not after date. In fact, the cunning little algorithm divested itself of all those bank

shares before the financial crisis of 2008 started. And a good thing too: LLOY.L shares plummeted from around £300 per share to under £30 during the recession.

With great power comes great responsibility

If you were to take £1,000 and invest it in a savings account in 1988, you'd need an annual interest rate of 10% to give returns comparable with those we've seen with these simple schemes. This may lead you to think that the author is very rich, only troubled by the choice of which volcano to hollow out for a secret base, or what colour boots his hordes of minions should wear. In truth though, I have not invested heavily in shares for two reasons. Firstly, because, like most people, I don't have enough money to risk losing it and so when I do have some money to spare, I place it in safe savings accounts with modest interest rates.

Secondly, simply speculating on the stock market has ethical concerns. Gambling on companies you know nothing about can, in concert with a mass of similarly ignorant or short-sighted investors, cause bubbles and busts, of which two are present in the data we've looked at – the dot-com bubble in 2000 and the Great Recession starting in 2008.

If you do wish to invest, you could do worse than follow the example of John Maynard Keynes, who advocated taking an interest in the companies and investing in those that would not just be profitable, but productive to the real economy too. That human touch, together with algorithms like those we've begun to develop here, can make for investments that are both profitable and ethical. 

Andrew Conway, millionaire philanthropist, tracks the stars to predict the future – just like real economists!

BUILD AN EMAIL SERVER – ROUNDUCUBE AND CYRUS

Send emails over a convenient webmail system without having Google’s prying eyes snooping on your communications.

WHY DO THIS?

- Access your email from anywhere.
- Keep giant corporations out of your inbox.
- Take back control of a vital service.

LV PRO TIP

You’ll find Roundcube’s installation instructions at `/usr/share/webapps/roundcubemail/INSTALL`.

LV PRO TIP

We covered installing and configuring the *Cyrus* IMAP server in issue 8.

On page 68 of Fosspicks this issue, Ben looks at *Roundcube*, the web-based email system that we’ve been using at Linux Voice Mansions for the last year-and-a-bit. *Roundcube* is a browser-based IMAP email client, and we’re going to show you how to sample its excellence. We’ll assume you already have a web server; we’re using *Apache* and we’ll configure a virtual domain to host webmail. First install *Roundcube*; it’s a PHP application, so those dependencies will also be installed. It also needs a database – we’ll use *SQLite* here and have included those dependencies too:

\$ pacman -S apache roundcubemail sqlite php-sqlite

You can use a different database if you want; *Roundcube* supports *MySQL/MariaDB*, *PostgreSQL* or even *MS-SQL*. Choose whatever suits your requirements.

The *Roundcube* configuration file is in `/etc/webapps/roundcubemail/config`. You need to make a copy of `config.inc.php.sample` as `config.inc.php` and edit it according to your needs. Make the following changes at least:

\$config[smtp_server] = 'localhost';

\$config[smtp_port] = 587;

\$config[db_dsnw] = 'sqlite:///var/cache/roundcubemail/sqlite.db?mode=0646';

\$config[enable_installer] = true;

This configures *Roundcube* to use our mailserver’s submission port so that outbound mail passes through our outbound content filter. The database setup is next; we’re using *SQLite* so just need to point to a location that is writable by *Roundcube* – the database will be created automatically. The last thing we do is enable the installer; we’ll use it shortly to verify our configuration and test the installation. But

first we need to configure the web server.

The *Roundcube* package includes an *Apache* configuration file that you can integrate into your *Apache* configuration. Copy it into place and link it into *Apache*’s main configuration:

\$ cp /etc/webapps/roundcubemail/apache.conf /etc/httpd/conf/extra/httpd-roundcube.conf

\$ echo "Include conf/extra/httpd-roundcube.conf" >> /etc/httpd/conf/httpd.conf

\$ mkdir /srv/http

Restart your webserver to make the changes take effect and then point a browser at `http://mailserver/roundcube/installer`. The first page checks the environment to make sure everything necessary is in place. Pay particular attention to make sure your desired database driver is available. The second page enables you to edit the configuration, but you have to download and save it over the existing one; you might find it easier to just use a text editor. The final screen verifies required write access to parts of the filesystem and allows you to test the IMAP and SMTP connections.

The last thing the installer does is remind you to disable it – you just need to remove the **enable** setting that we added to the configuration. Now you can point your browser at your webmail server and try it out. Log in as the test user that we created before (we used a username of “testuser” and password “testpass”).

Sieve it

We want to give our IMAP users some control over the delivery of their mail to enable them to forward it to someone else, file it into sub-folders or even reject it. The Sieve language enables end users to provide rules for the server to apply when delivering their mail. Rules are written using a simple language, are stored on the server and accessed via a daemon called **timsieved** that is part of the *Cyrus* IMAP server. You can use **telnet** to verify that you connect to it:

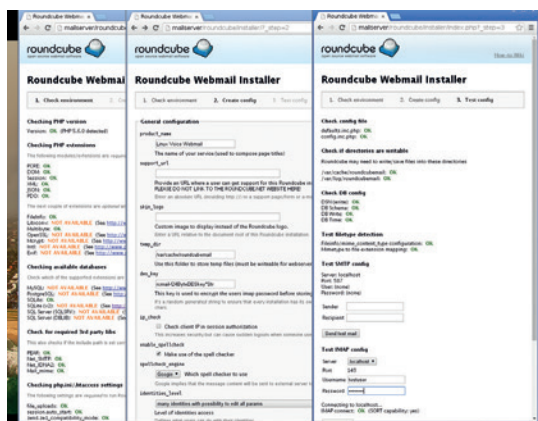
\$ telnet mailserver sieve

You wouldn’t normally do that though, because the *Cyrus* server comes with **sieveshell**, an FTP-like command-line tool for uploading, downloading, activating, de-activating and listing Sieve scripts. To use it as the current user:

\$ sieveshell mailserver

If your Linux user is different to your mail user then you’ll need to use the **--user** argument to supply it.

Roundcube’s installer helps make sure everything is set up properly, but don’t forget to disable it when you’re done!



sieveshell will request your IMAP password and display a prompt once it authenticates you. Enter **help** to see the available commands; there's also a man page with similar information.

Sieve scripts follow a basic syntax that enables a user to forward, discard or sort messages into sub-folders within their mailbox. The message header information is used to decide which action to take. The language is backed by an internet standard (RFC 5228) and you can find information online to get you up to speed. <http://bit.ly/sieve-tutorial> is a good one to start with. Here is an example:

```
require ["fileinto"];
if address :is "From" "bob@example.com"
{
    fileinto "Bob's Ramblings";
    stop;
}
```

Write that into a file, say **testuser.sieve** and use **sieveshell** to upload and then activate it:

```
$ ./sieveshell --user testuser mailserver
Please enter your password:
> put testuser.sieve
> activate testuser.sieve
> list
testuser.sieve <- active script
> quit
```

The daemon performs syntax checking and won't allow an erroneous file to be uploaded. Once it's in place, any mail from Bob will be filed into the sub-folder (if it exists or, otherwise, into the Inbox). Note that each user can keep many scripts on the server but only one of them can be active.

For Sieve to be of any use to our users, they need to be able to manage their own rules. You could give them their own copy of **sieveshell** but most users would prefer to manage their rules from within their own email client.

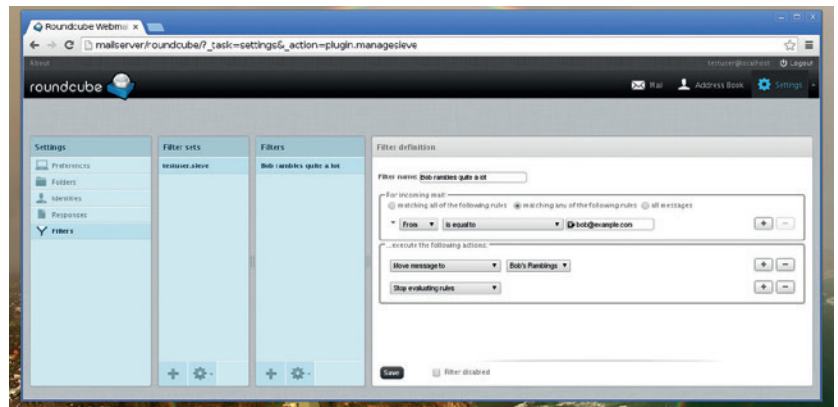
There is a Sieve add-on for *Mozilla Thunderbird*, but the current release (0.2.2) doesn't work with *Thunderbird* version 20 and above. You'll need to either wait for version 0.2.3 of the plugin or install a nightly build from GitHub (see <https://github.com/thsmi/sieve/tree/master/nightly>). Once you have a working

Sieveshell for command-line users

The **sieveshell** tool is part of the mail server but, by copying a few files, you can provide it to command-line savvy end-users so they can manage their *Sieve* rules from the command line. Copy the following files from the mail server onto any other machine where you would like **sieveshell** to be available:

```
/usr/bin/sieveshell
/usr/lib/perl5/site_perl/Cyrus/SIEVE/managesieve.pm
/usr/lib/perl5/site_perl/auto/Cyrus/SIEVE/managesieve/managesieve.so
/usr/share/man/man1/sieveshell.1.gz
/usr/share/man/man3/Cyrus::SIEVE::managesieve.3pm.gz
```

This does, of course, require Perl and binary compatibility and it isn't something that the *Cyrus* project officially supports. But it may work out for your command-line users.



version, it's easy to write Sieve scripts from within *Thunderbird*. You work directly on scripts stored on the server and you get immediate feedback when your scripts contain errors.

Roundcube comes with Sieve support included, but it's in a plugin and you have to enable it. First, set up its configuration by making a copy of the supplied example

```
$ cp /usr/share/webapps/roundcubemail/plugins/managesieve/config.inc.php(.dist,)
```

and then enable it by adding **managesieve** to the active plugins array defined in the main configuration file, **/usr/share/webapps/roundcubemail/config/config.inc.php**. Once it's enabled users will see a 'Filters' area on the *Roundcube* settings page that presents a user-friendly rule editor that doesn't require knowledge of the Sieve language.

Batten down the hatches...

To allow our users to access their mail remotely, we want to expose our server's IMAP and ESMTP interfaces to the internet. However, before doing this we need to implement TLS (Transport Layer Security) and mandate its use. And that requires a server certificate. We'll assume you have one, but our boxout shows how you can quickly make one that is good enough for testing.

To configure the IMAP server to use TLS, you need your private key, certificate and, if your certificate is signed by a certificate authority, its CA certificate. Store them somewhere that is accessible to both *Postfix* and *Cyrus*. You could place them in **/etc/mail**. Edit **/etc/cyrus/imapd.conf** to tell *Cyrus* where to find the key and certificates. Add these definitions (if you have a self-signed certificate then you can omit the last one):

```
tls_key_file: /path/to/private-key
tls_cert_file: /path/to/server-certificate
tls_ca_file: /path/to/ca-certificate
```

You also need to configure *Cyrus* to listen for secure IMAP connections. You can use **sed** to uncomment the **imaps** entry (or just edit the file):

```
$ sed -e '/^#\s*imaps\s/s/^#/' /etc/cyrus/cyrus.conf
```

You can also restrict insecure IMAP connections to the local machine (*Roundcube* will still connect this way):

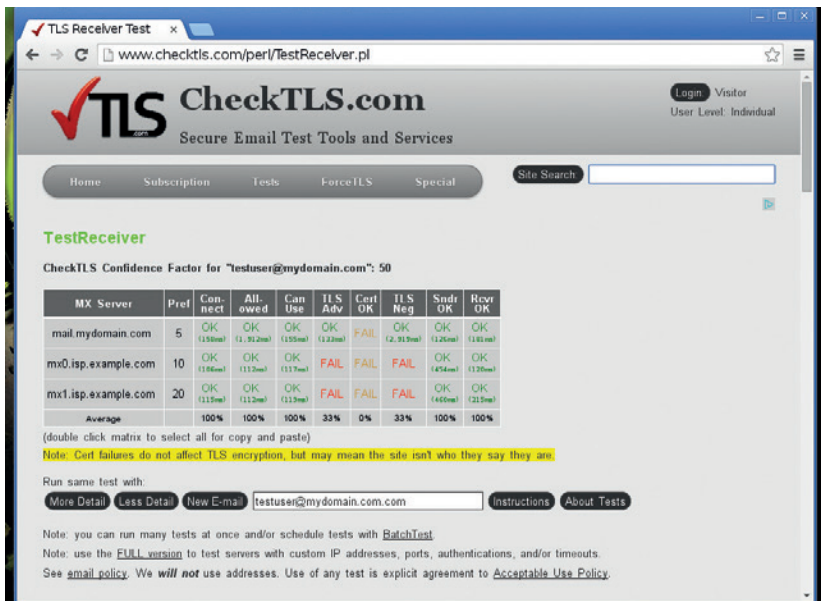
Roundcube's default Sieve implementation hides the script detail, making it a good choice for non-technical users. There are other alternatives available - see <http://plugins.roundcube.net>.

LV PRO TIP

php -m will confirm your PHP database modules are set up correctly.

LV PRO TIP

You don't have to implement webmail on the mail server - use a separate server if you want to. It might even be better that way!



You can check the TLS capabilities of your server. We got a certificate failure because we used a self-signed certificate. Our backup MX doesn't fare so well – a typical ISP!

```
$ sed -i -e 's/listen="imap"/listen="localhost:imap"/' /etc/cyrus/cyrus.conf
```

Finally, restart *Cyrus* to effect the changes:

```
$ systemctl restart cyrus-master
```

Configuring *Postfix* is similar. You need to add some configuration options to `/etc/postfix/main.cf`:

```
smtpd_tls_key_file = /path/to/private-key
smtpd_tls_cert_file = /path/to/server-certificate
smtpd_tls_CAfile = /path/to/ca-certificate
smtpd_tls_security_level = may
smtp_tls_security_level = may
```

There are two groups of settings that apply to the smtp client-side (for outgoing mail) and the `smtpd` server-side (for inbound mail). We specify the same key and server certificate that we used for *Cyrus*.

The `may` security level is what enables TLS and is one of several possible levels. It announces STARTTLS support to remote SMTP clients but doesn't require encryption to be used because the SMTP over TLS specification requires that publicly accessible servers (those referenced by public MX DNS records) do not enforce TLS.

However, you can enforce TLS on the submission interface so that email clients must encrypt, by overriding the security level for the `submission` service defined in `/etc/postfix/master.cf`:

```
-o smtpd_tls_security_level=encrypt
```

Another security measure that you can take is to require your clients to authenticate with the server before they can send email through it. You can achieve this by configuring the submission service to use the same SASL authentication service as the IMAP service. A few changes to its configuration in `/etc/postfix/master.cf` makes this happen:

```
-o smtpd_sasl_auth_enable=yes
-o smtpd_recipient_restrictions=permit_mynetworks,permit_sasl_authenticated,reject_unauth_destination
```

The `recipient` restrictions control who can use the service to send mail. You must also add `permit_sasl_authenticated` into the existing `smtpd_client_`

restrictions before the `reject` restriction. The client restrictions control who can connect to the service.

These changes allow clients outside of `mynetworks` to connect to and send email if they can authenticate, but we need to define what `mynetworks` is. This is a list of trusted SMTP clients – those allowed to relay (send) email. *Postfix* determines this automatically using the machine's network interfaces, trusting anything on the same network. We don't want that; we want all clients to authenticate. To achieve this we can explicitly specify in `main.cf` to only trust the local host.

```
mynetworks_style = host
```

Trusting the local host will allow our webmail system to send mail without going through the authentication mechanism.

There is one other thing to do, and that's to configure the SASL library that *Postfix* uses. The library looks in the `/usr/lib/sasl2` directory for application-specific configuration files that are named after the application. In *Postfix*'s case, each service has its own file – the SMTP server's is called `smtpd.conf` (this is a default value that can be overridden with a *Postfix* configuration option, but we'll just go with the default). Because we just want *Postfix* to authenticate in exactly the same way as the IMAP server, we can extract the required settings with a little bit of `sed`:

```
$ sed -n -e 's/^sasl_(.*)/1/p' /etc/cyrus/imapd.conf > /usr/lib/sasl2/smtpd.conf
```

With the changes made, you can restart the services:

```
$ systemctl restart saslauthd postfix
```

From now on, when users send mail, they will need to connect using TLS and supply their username and password. It's important to understand what TLS gives us. It's Transport Layer Security; that means it protects the connection between the mail client and the mail server that is made for each message that a user sends. The server will also attempt to use TLS when relaying those messages to their destinations, but not all mail systems support it, in which case the transmission falls back into the clear.

Regardless of whether TLS was used, messages aren't encrypted once delivered. There is nothing stopping them from being read, forwarded or altered by anyone able to access them legitimately or otherwise. We can further protect messages with

A server certificate

Enter the commands below. When asked for a password, use "testpass" or anything else that you wish. Of all the other questions, the important one requests a Common Name. Enter the server's fully-qualified domain, `mail.mydomain.com`.

```
$ openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out server.pem
$ openssl req -new -x509 -key server.pem -subj "/CN=$(hostname -f)" >> server.pem
$ chmod 600 server.pem
```

This creates a private key and a self-signed certificate, both in the same `server.pem` file. You can specify this same file for both your private key and certificate. Although self-signed certificates work, they will annoy your users!

content signing and encryption. When a sender signs a message, its recipients can verify that it is authentic and has not been altered. When a sender encrypts one, it is only readable by its intended recipients.

Both signing and encryption use asymmetric cryptography using public and private key-pairs and there are two implementations in common use for email: S/MIME uses X.509 certificates in a similar way to SSL websites, whereas OpenPGP implements a different key system based on webs of trust where entities known to each other sign each other's keys. While corporate entities might prefer the S/MIME approach, the fact that it requires trusting a potentially unknown certificate authority to sign keys (in contrast to OpenPGP's approach being decentralised) and that PGP has been around longer makes PGP the more popular choice for email.

PGP also sees other uses in the open-source world where, for example, it's used to sign software releases. The open-source implementation is called *GNU Privacy Guard*, or just *GnuPG*. You can check if you have it installed with `gpg --version` and, if necessary, install it on Arch with `pacman -S gnupg`.

Encryption is typically a user-focussed activity: When Alice sends a secret message to Bob, she uses his public key to encrypt it and he must use his private key to decrypt it. If Alice wants to sign the message, she uses her private key and Bob can use her public key if he wants to validate its authenticity. Neither Alice nor Bob trust their mail systems' administrators, so they keep their private keys securely on their own computers and use them with their email client.

Enigmail is an add-on for *Mozilla Thunderbird* that provides OpenPGP tools that use *GnuPG* underneath. You install it using *Thunderbird's* Add-Ons facility and follow the setup wizard that it launches. You can either use an existing key or create a new one. Once it's installed, you'll have a new *Enigmail* menu option. You can try sending signed and encrypted email using Adele, the Friendly OpenPGP Email Robot: just send a message to `adele-en@gnupp.de` and you should receive a response showing that your test could be decrypted and/or signed. You need to attach your public key, but don't encrypt the attachment otherwise the robot won't work.

You can also look at supporting encryption with *Roundcube*, but there isn't a complete out-of-the-box solution like *Enigmail* for it. You could investigate `rc_openpgpjs` or the Web Encryption Extension (<http://senderek.ie/wee>). Google is your friend here.

You can, however, use *GnuPG* on the server side if you want to. To demonstrate this, we'll configure our *Procmail* outbound filter to sign and/or encrypt outbound messages. To achieve this, we'll give the server its own private key for signing and use recipients' public keys for encryption. You can generate a key on the server; it needs to be owned by the user that runs the outbound filter (in our example this is the `cyrus` user). Press Enter when prompted for a passphrase to create a key without one; this is more

Apache quick start

If you haven't got *Apache* and just want to get it up and running to follow along with our *Roundcube* setup, here's what you need to do. Start out by installing *Apache* and *SQLite*, plus the necessary PHP modules. Then install the module into *Apache's* configuration:

```
$ pacman -S apache sqlite php-apache
php-sqlite
$ sed -i -e 's:^LoadModule mpm_event_module
modules/mod_mpm_event.so:LoadModule
mpm_prefork_module modules/mod_mpm_
prefork.so: \'
-e '/LoadModule dir_module/a LoadModule
php5_module modules/libphp5.so' /etc/httpd/
conf/httpd.conf
```

PHP also has a configuration file; you need to set a time zone and also load the

SQLite driver:

```
$ sed -i -e "s:::(date.timezone =)\$:\1 Europe/
London:" \
-e "s:::(extension=pdo_sqlite.so):\1:" /
etc/php/php.ini
$ echo "<?php phpinfo(); ?>" > /srv/http/
phptest.php
```

We also wrote a little test page so that we can check everything is OK, but you must start *Apache* first, which you do with `systemd`:

```
$ systemctl enable apache
$ systemctl start apache
```

and then point a browser at it. You should see something like the picture.

Further information is available on the Arch Linux wiki; see <https://wiki.archlinux.org/index.php/LAMP#PHP>.

appropriate for server use.

\$ gpg --gen-key

You should give the key a Real Name that is the same as the system user (`cyrus`) so that the key can be found. Otherwise you'll need to specify it in the following commands by appending the `--local-user` argument that specifies the key's ID.

The actual encryption and signing is done by the `/etc/procmail/outbound.rc` filter. Add a recipe for signed encryption and, in case that fails (as will happen if the recipient's public key is not in the key ring), another to sign it instead:

Encrypt for recipient and sign with server's key

```
:0 fbw
```

```
| cat | gpg --encrypt --armor --recipient $RECIPIENT --sign
```

If unable to encrypt, just sign with server's key

```
:0 fbwe
```


```
| cat | gpg --clearsign
```

You also need to add `RECIPIENT=${recipient}` to the `procmail-outbound` definition in `/etc/postfix/master.cf` and load any public keys for any recipients that you want to encrypt for.

This is a very basic server-side example to demonstrate how you could implement an outbound signing policy, but think of it as a starting for your own rules that suit your own requirements.

Where next...

There's plenty more that you can do to continue growing your mail system. Here are some suggestions:

- Implement a SASL back-end to authenticate from LDAP, perhaps your Active Directory server implemented with Samba.
- Implement a second server to use as a backup and use Cyrus-IMAP's replication capabilities to keep them in sync. 

John Lane provides technical solutions to business problems. He has yet to find something that Linux can't solve.

FORTRAN: CODING FOR SCIENTISTS, BY SCIENTISTS

Come back through time to the days of FORTRAN – the language of fluid dynamics, computational physics and more.

FORTRAN (it dropped the caps in 1990) is the oldest high-level language still written today. It's now over 55 years old and still in widespread use in the sciences, in high-performance computing, and in supercomputers. Its real strength is in numerical computation and complicated mathematical models (making it also popular in finance); and its position is hard to assail given the vast Fortran code library of numerical computation routines that's available. There are even people still using fixed-format F77 (see below), although most modern users have shifted to the easier free-format. It's probably not your language of choice for shiny Web 2.0 development, but it's fascinating to have a look at something with such a venerable and successful history.

History

John Backus, at IBM, proposed developing Fortran in 1953, with the aim of producing a more practical alternative to assembly language. Assembly language (now as then) uses mnemonics like ADD to describe basic operations, which are then translated into machine code (as you'll know from the assembly tutorial on page 106). This is one up from programming directly in machine code (as in the very earliest computers), but is still massively time-consuming and makes bugs very difficult to find. Backus' idea was to create a program that could turn something like mathematical notation into machine code. This wasn't a totally novel idea. UNIVAC had Short Code, which did something similar but very slowly around 1949, and Grace Hopper at UNIVAC was proposing a similar idea to Backus' at about the

same time (resulting in FLOW-MATIC, an ancestor of COBOL). But in 1953, there was nothing that really looked like a useful, functional, high-level language. IBM gave Backus the go-ahead.

Backus' team had completed a draft specification by 1954, and the first compiler was delivered in April 1957. An important aspect was that it was an optimising compiler; the language would (understandably) never be popular if it couldn't compete on performance with hand-coded assembler, which had been the main problem with Short Code. FORTRAN's focus was on numerical computation, as that was the chief use for IBM's machines at the time. It caught on extremely quickly, making it the first successful high-level language (that is, a language which is at least to some extent removed from the details of machine code).

The major advantage, and the reason for its popularity, was that programs could now be written much more quickly. Although FORTRAN was initially very limited in scope, it enabled the programmer to think in terms of algorithms, without then having to translate that by hand into machine instructions. Instead, the compiler did that hard work – and did it well. Its rapid spread led to manufacturers developing compilers for many different computers, making it arguably the first cross-platform language. This in turn helped popularise it further, because FORTRAN programs written for one computer could be run on another computer; something that just wasn't possible with assembler.

This first version of FORTRAN had 32 statements, including flow control (**IF**, **GOTO**, etc), I/O, and assignment statements. It was stored on punchcards, with one card per line of code. FORTRAN II, in 1958, allowed the programmer to create subroutines and functions, including passing parameters by reference, and global (or **COMMON**) variables. Subroutines could not however be recursive, as computers at the time didn't have a 'stack' concept. FORTRAN III was never released, and FORTRAN IV, begun in 1961, had a few minor improvements.

FORTRAN 66 was the next big step, providing an 'industry-standard' version of the language, based on FORTRAN IV. They also defined Basic FORTRAN, which removed all machine-dependent features (ie anything that relied on a particular brand of computer to work). Afterwards, however, compilers with various extensions were released to take advantage of other

```

juli@inspiral: ~/coding/fortran
File Edit View Search Preferences Tabs Help
1. juli@inspiral: ~/coding/fortran
juli@inspiral:~/coding/fortran$ f95 -o hello hello.f95
juli@inspiral:~/coding/fortran$ ./hello
Hello World
juli@inspiral:~/coding/fortran$

hello.f95 (~/.coding/fortran)
File Edit Tools Syntax Buffers Window Help
! Hello World
program hello
print *, 'Hello World'
end program hello

```

Hello there! Check out the user input boxout below and see if you can improve this code to ask your name.

possible features, which led the ANSI committee to revise the standard again. This took nearly a decade, but the update was finally circulated in 1977, resulting in FORTRAN 77, perhaps the most historically important dialect. Unlike earlier versions, F77 code is still running out in the world, and FORTRAN compilers now available will compile F77 as well as later versions. It had improved support for structured programming, now the norm in programming, much improved character input/output support (previously characters were unsupported, and had to be placed into numeric variables via the Hollerith constants technique, which was not very portable), support for lexical string comparison (demanded by the US Dept of Defence), and various loop extensions. F77 was the "Standard FORTRAN" for nearly fifteen years.

The next major rewrite, after many delays, was Fortran 90 (caps now abandoned). (For more on the delays and the byzantine behind-the-scenes wrangling, see Brian Meek's somewhat world-weary piece *The Fortran Saga* (<https://www.fortran.com/forsaga.html>.) The biggest change was that free-form source input was now allowed (see the section on fixed-form source). Array operations were also finally added, as was recursion, operator overloading, dynamic memory allocation, and much improved data structure handling. Modules were also now available, improving program structure and reflecting new programming practice since 1977. However, all F77 features were retained, and any compliant F77 program should also be a compliant F90 program. Fortran 95 was a minor revision, although it deleted a few features (dating back to pre-F77) that had been labelled obsolete in Fortran 90.

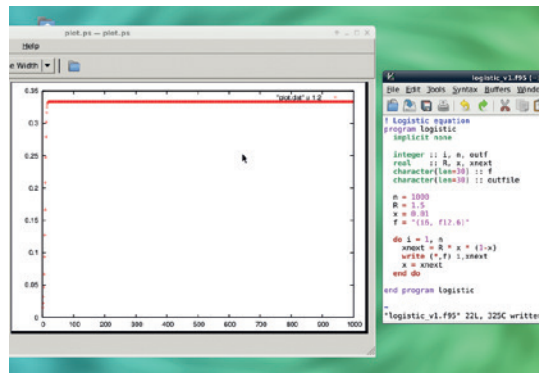
Another major revision has since been published, Fortran 2003 (and a minor update in 2008). This includes, among other things, object-oriented programming support, improved floating-point handling, and improved I/O. However, if you want to learn Fortran today, it's usually advised that you start with 90/95, and then go on to learn about the new features added by 2003. The rest of this article will use Fortran 95.

Hello World

To get started with Fortran, the easiest compiler to get hold of is the GNU compiler *GFortran*. Install the **gfortran** package for your distribution, or download it from the GNU website and compile from source. Other compilers include *G95* (free) and *NAG* (paid). Some compilers include extensions beyond the Fortran standard, which can be handy but equally lock you into that compiler; here we'll stick with the official standard.

You can run *GFortran* with the command **gfortran** or **f95**; I'll use **f95**. Once you've got the compiler installed, save this as **hello.f95**):

```
! Hello World
program hello
print *, "Hello World"
```



The population graph in our converges pretty quickly here, as per that long boring line.

end program hello

Compile it with **f95 -o hello hello.f95. -o hello** specifies the output file; without this option, the default is to compile to **a.out**. Run it with **./hello** and you should get the traditional Hello World output.

Looking at the source, you can see that comments start with **!**, and that programs begin with **program <name>** and end with **end program <name>**. **print *** means print to screen (print), with the appropriate format for the output (*). Both double and single quotes can be used from Fortran 90 onwards (single only for FORTRAN 77). Indentation isn't required in free format source code, but it does make for code that's easier to read.

Fixed Format and Free Format

Prior to Fortran90, Fortran code had to be written in fixed format. Initially, this was because FORTRAN in its earliest days was written on punch cards, and a specific format of card was expected. The basic rules looked like this:

- Maximum line length 72 characters. To continue a line, put any character in column 6 on the next (continuation) line.
- The first 6 columns must be empty (so each line starts with 6 spaces), unless the line is a continuation line or a comment.
- Comment lines have ***** or **c** in the first column.
- Spaces are ignored altogether. **endprogram** and **end program** are read exactly the same.

So the "Hello World" program above would look like this in fixed format:

```
c Hello World
program hello
  print *, 'Hello World'
end program hello
```

This is, as you will readily understand, a bit of a faff. Happily, as of Fortran 90, free format was introduced; though fixed format is still understood by Fortran compilers, to make sure old code could still be used. Free format looks a lot more like any other modern language (all the rest of the code in this tutorial is in free format), but there are still some rules:

- Lines can be up to 132 characters long; to continue a line use **&** at the end of the first line.
- If the split is in the middle of a name, use another **&** at the start of the next line:

LV PRO TIP

Structured programming argues that programs are made up of sequences, selections (**if/then**), and iterations (**while**, **for**, **do**), with blocks and subroutines to group statements together. This is in contrast to unstructured programming, which simply has a sequence of commands.

User input

You could set up the program to ask you for a specific version of *x* and *R*. Add these lines:

```
integer :: outf
character(len=30) :: outfile
outf = 3
outfile = "plot_v1.dat"
print *, "Please enter an initial x value between 0 and 1"
read *, x
print *, "Please enter an initial R value greater than 1"
read *, R
open (unit=outf, file=outfile, action="write", status="replace")
```

! Do loop and other variables as before, but alter the write line:

```
write (outf, f) i, xnext
outf and outfile are used to write to a specific file, rather than specifying the file on the command line. This is because if you use > to redirect on the command line, the print lines will also be redirected. You won't see them, and they'll cause problems in the data file. With open, as here, you need to provide an integer filehandle (should be greater than 3, as 0, 1, and 2 are the system filehandles; and Fortran itself reserves 5 and 6), a string filename, and options: here, the write action, and replace rather than append status.
```

```
character :: longname*100
longname = "this is a terribly long line, so you need an ampersand here & &and another one here"
```

- Comments start with !.
- Whitespace doesn't matter in some places: ENDIF and end if are treated the same, and indentation is not significant. However, you can't have embedded spaces in variable names or in numbers, as you could in earlier FORTRAN.

- You can put multiple statements on one line by separating them with ; (although the next person reading your code may or may not thank you for it). Most compilers assume free format if the source filename ends in .f95 or .f90, and fixed format if it is .for. You can also specify -ffixed-form or -ffree-form on the Linux command line when invoking the GCC compiler.

The logistic equation

The logistic equation is an example of a very basic population model that shows some distinctive characteristics of feedback systems. It can be defined in a couple of ways, but we'll use this one:

$$P_{i+1} = R * P_i * (1 - P_i)$$

P_i is the population at generation *i*. *R* is a growth rate term; you could think of it as representing the resources available. A more basic version of this equation would be $P_{i+1} = R * P_i$, in which the population at generation *i+1* is purely dependent on the population at generation *i* plus this growth rate. A system with better resources would grow faster. The $(1 - P_i)$ term is added to represent the idea that as

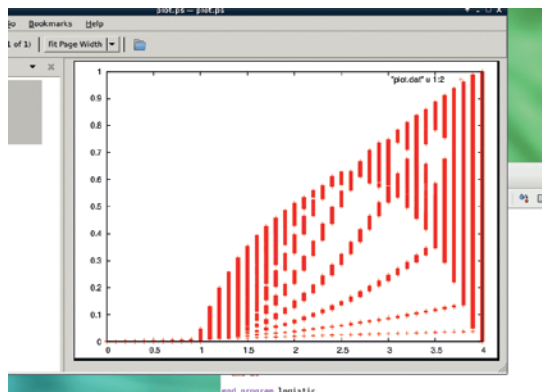
populations grow larger, expansion becomes harder. What we might expect is that over term, the value converges, and eventually *P_i* and *P_{i+1}* become the same.

So here's a first go at an iterative program that looks at what does happen to population over time (specifically, over 1,000 generations). Save this as **logistic.f95**:

```
! Logistic equation
program logistic
implicit none
integer :: i, n
real :: R, x, xnext
character(len=30) :: f
n = 1000
R = 1.5
x = 0.01
f = "(i6, f12.6)"
do i = 1, n
xnext = R * x * (1-x)
write (*,f) i, xnext
x = xnext
end do
end program logistic
```

Any line beginning **!** is a comment. **implicit none** is an inheritance from FORTRAN 77 and earlier. Originally, Fortran had implicit variable typing, done by variable name. Variables beginning **i, j, k, l, m,** or **n** were taken to be integers (since scientists and mathematicians – and Fortran was written by scientists – expect **i** and **j** to be integers), and all other names were real numbers, unless explicitly declared to be strings or characters. Due to the desire for backwards compatibility, this had to still work in F90 and onwards; but it's not a good modern programming practice. The solution was **implicit none**, which turns off implicit typing and requires all variables to be explicitly declared.

The next three lines do the declaring, creating integers, real numbers (real has 6 decimal digits of precision), and a string, which is declared as a character array of length 30. Other basic data types are **double precision** (13 digits of precision), **logical** (true/false, like a Boolean variable), and **complex** (complex numbers). Fortran 90 also introduced derived types so you can create your own types. Note



The interesting stuff is as *R* gets bigger, especially between about 3 and 4.

that we are specifying the initial value of **P0** (**x = 0.01**) as well as **R**.

The string **f** will be a formatting line for the output; the format here is (**i6, f12.6**). **i6** is an integer format, requiring up to 6 positions (ie it could output a number up to 999999, or -99999 as the negative sign takes up an output position). If you tried to output 1000000 it would print ********* to warn you that the number was too large for the format string. **f12.6** is for real numbers, and indicate 12 positions, of which 1 will be the decimal point, and 6 will be to the right of the decimal point. So you could print up to 99999.999999. If there are more than 6 numbers after the decimal point it will be rounded appropriately. Again, if the whole number is too large, it prints asterisks.

The main body of the program is the **do** loop, which loops for **i** between 1 and **n** (here 1000). It calculates **xnext** (**xi+1** in the equation) from **x** (**xi**), outputs **i** and **xnext** to screen, then stores the calculated value in **x** for the next loop. Compile and run as follows:

```
f95 -o logistic logistic.f95
```

```
./logistic > plot.dat
```

This saves the output as a file, but it's just a file of numbers. To view it as a plot, you'll need to use *Gnuplot* and *Evince* (or another PostScript viewer). Save this as **plot.gp**:

```
set term postscript enh color
```

```
set output "plot.ps"
```

```
plot "plot.dat" u 1:2
```

The details of *Gnuplot* are outside the scope of this tutorial, but basically this sets up your terminal, sets the output filename, and tells *Gnuplot* to plot the contents of **plot.dat** using the first two columns as **x** and **y** axes. Generate and view the plot with

```
gnuplot plot.gp
```

```
evince plot.ps
```

If you look at the output file **plot.dat** you'll see that the equation rapidly converges to 0.333333. This suggests that a stable value for the population is 0.333333. This might be the same as the carrying capacity of the environment (ie the population level that the environment can support), but it might also be lower. Try changing the value of **x** to see if the start size of the population has an effect; and **R** to look at what effect the growth rate has.

Multiple loops

Here's another version, with multiple loops. This will allow us to look at what happens to the convergence point with different values of **R** (**x** here stays the same).

```
! Logistic equation
```

```
program logistic
```

```
implicit none
```

```
integer :: i, j, m, n
```

```
real :: R, x, xnext
```

```
character(len=30) :: f
```

```
m = 4000
```

```
n = 1000
```

```
f = "(i6, f12.6)"
```

```
do i = 1, m
```

```

juli@inspiral:~/coding/fortran
File Edit View Search Preferences Tags Help
1. juli@inspiral:~/coding/fortran
juli@inspiral:~/coding/fortran$ time ./logistic > plot.dat
real    0m4.394s
user    0m4.132s
sys     0m0.196s
juli@inspiral:~/coding/fortran$ f95 -o logistic logistic.f95
juli@inspiral:~/coding/fortran$ time ./logistic > plot2.dat
real    0m10.387s
user    0m9.369s
sys     0m0.312s
juli@inspiral:~/coding/fortran$ ls -l plot*.dat
-rw-r--r-- 1 juli@inspiral 68000000 Feb  5 15:28 plot.dat
-rw-r--r-- 1 juli@inspiral 30583765 Feb  5 15:28 plot2.dat
juli@inspiral:~/coding/fortran$ ls -lh plot*.dat
-rw-r--r-- 1 juli@inspiral 65M Feb  5 15:28 plot2.dat
-rw-r--r-- 1 juli@inspiral 30M Feb  5 15:28 plot.dat
juli@inspiral:~/coding/fortran$

```

```
R = i
```

```
R = R/1000
```

```
x = 0.01
```

```
do j = 1, n
```

```
  xnext = R * x * (1-x)
```

```
  write (*,f) R, xnext
```

```
  x = xnext
```


```
end do
```

```
end do
```

```
end program logistic
```

The meat of the program is still the same (that inner **do** loop, now labelled with **j**). But it now runs once for each value of **R** between 1 and 4. Since **i** varies between 1 and 4000, and **R** is defined as 1/1000 of **i**, it jumps in 0.001 steps. Note that if you try to define **R = i/1000** it won't work; dividing an integer always results in an integer. Instead, define **R** as **i** (translating it into a real number) then divide it by 1000.

Because of the way this is plotted, the long thick lines for **R** between 1 and 1.5 actually represent the population converging to a specific level (the top of the line, which is the population stable point for that value of **R** and **x**. (You could edit the code to look for the convergence point and only plot that, if it exists.) As **R** gets bigger, the graph gets more complicated. At some values there seem to be multiple convergence points; towards the far right of the graph, what we're seeing is actually a chaotic effect.

A couple of potential improvements would be to try changing the initial value of **x** to see what happens at different **R** values; and to test for a convergence point and to plot only that convergence point if it exists, rather than all the points that lead it there. This could make the graph a bit clearer. But this initial graph should be enough to demonstrate the possibilities of feedback equations like this. If this sort of numerical modelling is your bag, you can find sample Fortran programs online for nearly anything you want to do. There's extensive documentation available, although Fortran programs can be a bit of a nightmare to debug as error messages are not always readily comprehensible. Try it out, and enjoy that sense of hacker companionship linking you to those 1950s IBM pioneers who paved the way for the vast field of modern languages. 

Juliet Kemp is a scary polymath, and is the author of *Apress's Linux System Administration Recipes*.

Time is a useful utility for a quick benchmark of something like this.

BEN EVERARD

PYTHON: CODE REUSE AND NOTIFICATIONS

Don't keep writing the same code over and over again – put it in a package and reuse it.

WHY DO THIS?

- Get your applications to remind you to take a break. Or work harder.
- Save time typing the same lines of code over and over again.

In this month's code ninja, we're going to take a look at two different, but important topics: desktop notifications and code reuse. Desktop notifications are a great way for your code to let the user know that something has happened. They work in a standard way across almost all Linux desktops, so the code here isn't tied to a particular widget set.

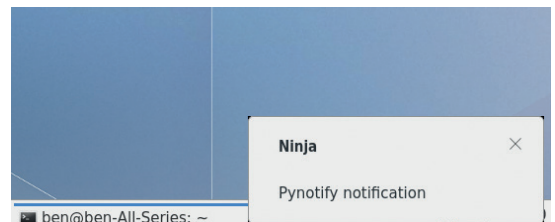
There are libraries for doing this in just about every popular language. In Python, the most popular library for this is **pynotify**. You can test it out with the following script that should pop up a message:

```
import pynotify
pynotify.init("test")
notification = pynotify.Notification("Hello")
notification.show()
```

While all desktops should be able to display messages, they each have different capabilities. You can find out what capabilities your desktop has using the **pynotify.get_server_caps()** function. This returns a list of the various things the desktop can display. For example, if the server can display hyperlinks (like the links on a web page), then one of the entries in the list will be the string **"body-hyperlinks"**.

The following code could be used to send someone to the Linux Voice website:

```
import pynotify
pynotify.init("LinuxVoice")
capabilities = pynotify.get_server_caps()
if 'body-hyperlinks' in capabilities:
    message = "<a href='http://www.linuxvoice.com'>Mag Website</a>"
else:
```



The notifications will look different depending on your desktop environment. This is how they appear on *LXQt*.

```
message = "Point your web browser to www.linuxvoice.com"
notification = pynotify.Notification(message)
notification.show()
```

Save electrons

This brings us onto the second aspect of this month's ninja: code reuse. As you can see, the code to pop up a notification is only a few lines, but why retype it every time you want to send a message? Instead, we can put it all into functions, and then we need only call the functions when we want to display a message. We can use the following functions to display messages and find out if we can include hyperlinks:

```
import pynotify
def msg(text):
    notification = pynotify.Notification(text)
    notification.set_urgency(pynotify.URGENCY_NORMAL)
    notification.show()
def hyperlink():
    capabilities = pynotify.get_server_caps()
    if 'body-hyperlinks' in capabilities:
```

Distribution Sharing your project with the world

You have probably noticed that when you install Python packages, you don't just copy directories into your Python path manually. Instead there are tools that do this (and provide other services as well). These tools are themselves organised in a package called **setuptools** (you'll need to install this). To use this, first we need to put our package directory in another directory of the same name, so we have **simpledesktop/simpledesktop**. Then, in the first **simpledesktop** directory, create a file called **setup.py**. This should have the contents:

```
from setuptools import setup
setup(name='simpledesktop',
      version='0.1',
      description='easy desktop interaction',
      author='Ben Everard',
```

```
author_email='ben@linuxvoice.com',
      license='GPLv3',
      packages=['simpledesktop'])
```

Setuptools handles everything to do with putting your package in the right place. Once you've created this, you just need to open a terminal in the same place and run:

```
sudo python setup.py install
```

You should then be able to access the package from any Python session on the computer. If you wanted to share a package with other people, you can share it in this format and let them run **setup.py**. However, if more than a few people are likely to want to use it, a better option is to register it on the Python Package Index (PyPI). This can be done using the **setup.py** script. Full details are in the Python documentation at <https://docs.python.org/2/distutils/packageindex.html>.


```

return True
else:
return False

```

If you include the above section of code into a file, you can then send messages with:

```

pynotify.init("test")
if hyperlink:
    msg("hyperlinks work")
else:
    msg("hyperlinks don't work")

```

This makes it easier if you need to send several notifications from a single program, as you can just call the functions, but why should you have to include these functions in every program that you want to use notifications? You don't! Instead you can create a module that you import (in a similar way you imported **pynotify**) that just makes the functions available.

There's nothing special about modules in Python. They're just regular Python files that are called by name. We'll call our module **noti**, and this means we'll use the filename **noti.py**. Create this file and include the code:

```

import pynotify
def set_app(title):
    pynotify.init(title)

```

You'll also need to include the **msg** and **hyperlink** functions from above. If you save this in a directory, then any time you start Python in the same directory, you can use **import noti** to bring in all the functions, eg:

```

import noti
noti.set_app("test")
if noti.hyperlink:
    noti.msg("<a href='http://www.linuxvoice.com'>website</a>")
else:
    noti.msg("no hyperlinks")

```

This works because any time you use the **import** command, Python looks for the appropriate file to import. There's a particular set of directories it looks in to find the file, and that's stored in **sys.path**. You can view this with:

```

import sys
print sys.path

```

If you put your code in any directory in that path (which should include the current directory), then the **import** command will find it.

Packing it up

This situation works well for simple modules, but sometimes it's not convenient to put everything in a single file. Suppose, for example, we wanted to extend our simple notifications tool to include alert popup boxes as well as desktop notifications, it would make sense to put that in a different text file. We'll do just this by creating another module that's a simple wrapper around *EasyGUI* (you'll need to install this from your package manager):

```

import easygui
def alert(text):
    easygui.msgbox(text, title="alert")

```



PyPI (<https://pypi.python.org/pypi>) is the ultimate source of Python packages and can fulfil just about every need.

If you save this in a file called **alert.py**, you can import it in the same way as **noti.py**, and use the alert function with:

```

import alert
alert.alert("Hello World!")

```

We can bring our two modules together to make a package for simple desktop interactions. Packages are similar to modules in that they can be imported, but they're different in that they contain multiple files that can work together or separately.

Packages are directories that include a file called **__init__.py**. This file can be empty, in which case it just serves to show Python that the directory includes a package, or it can include any code that needs to be run when the package is first imported.

We'll call our package **simpledesktop**, so you'll need to create a directory called **simpledesktop**. Inside that directory, you'll need the **alert.py** and **noti.py** files that we created earlier. You'll also need a file called **__init__.py**. Since we don't need any initialisation code, this can be empty.

Now, from the directory in which **simpledesktop** sits (so, for example, if you created **simpledesktop** in your home directory, you'll need to run this in your home directory), you can run Python interactively and do the following:

```

>>> from simpledesktop import alert
>>> alert.alert("Hello World!")

```

The **__init__.py** file does have one useful feature we can use. If it has a variable called **__all__**, Python will use that to decide which parts to import if the user does **from simpledesktop import ***. In our case, you need to add the following line to **__init__.py** to bring in everything:

```

__all__ = ["alert", "noti"]


```

You can now (in a new Python session) run:

```

from simpledesktop import *
alert.alert("Hello World")
noti.set_app("Hello")
noti.msg("World")

```

Packages can be added anywhere in the path and Python will find them (as long as the **__init__.py** file is there). Using this structure, you can create packages as complex as you like, and you should never have to re-type commonly used code again. 

ASMSCHOOL: GETTING DOWN TO THE BARE METAL

MIKE SAUNDERS

Part 3: It's time to say goodbye to the operating system, and boot your PC from your own code.

WHY DO THIS?

- Learn what compilers do behind the scenes
- Understand the language of CPUs
- Fine-tune your code for better performance

In the last two issues we've gone through the basics of assembly language, looking at registers, loops, conditionals, the stack and other topics. You now have enough knowledge to write simple assembly programs on Linux – but we're going to get even more low-level this issue. Yes, we're going to jettison the operating system and get down to the bare metal of your PC. You'll write code that executes directly on the CPU and has full control of the

machine, without the operating system interfering in any way. Exciting times!

To do this, you need to understand how the PC boot process works, so we'll go through that step by step. Then we'll create a simple bootloader that outputs a message to the screen, and show you how to run it in an emulator. We'll also make it write to removable media such as a USB key, so you can try it on real machines and win an insane number of geek points.

1 THE X86 PC BOOT PROCESS

When you hit the power button on your PC, a bunch of things happen before the Linux kernel is loaded into your RAM banks and executed. Indeed, the PC is just a bunch of chips and has no idea of what a kernel is,

or where to find it, or how to even read the filesystem on the disk. A PC on its own would be useless, but fortunately almost every PC includes a BIOS – a “basic input/

“Fortunately almost every PC includes a BIOS – a ‘basic input/output system’.”

output system”. (Some very recent PCs include an emulated BIOS, or have deprecated it in favour of the alternative UEFI method – so if you only have UEFI-equipped PCs, you'll need to use a PC emulator for this tutorial, as explained later.)

Here's our code, running in a PC emulator – no operating system required!

The BIOS is simply some firmware provided in the PC, and contains software that the CPU executes as soon as the PC is turned on. Typically the BIOS will perform a bunch of checks to make sure that the PC is in a sane state – for instance, to check that RAM banks are present, and to produce the classic “Press F1 to continue” message when you don't have a keyboard plugged in.

The BIOS will then attempt to load a chunk of data from some form of media. Most BIOSes know how to access floppy disks, hard drives and CD/DVD-ROM drives, and sometimes USB keys as well. But BIOSes are small, and don't have space for lots of filesystem drivers. So the BIOS doesn't understand the ext4 or Btrfs filesystems as used on Linux, and therefore can't navigate a partition to find the Linux kernel, but it can grab the first 512 bytes from the drive, load it into memory and execute it.

Multi-stage to orbit

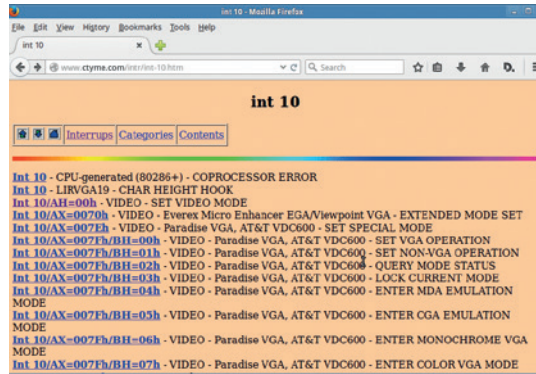
You can't do much in 512 bytes, but this chunk of code (known as the first stage bootloader) typically has enough logic to load more data from the disk, this time several kilobytes, which can provide a more fully-featured bootloader with menus and options. Alternatively, this code may go on to load more data from the disk and present an even more advanced bootloader with graphics and wider filesystem support. So in the PC boot process, the computer “pulls itself up by its bootstraps” (which is where the term “booting” comes from).

Now, we can write our own code to fit into these 512 bytes and have full control over the machine. But you may be wondering: without an operating system, how are we going to make a message appear on the screen? Won't we have to write a complicated video



driver, with pixel-plotting routines and font definitions, which will surely be much larger than 512 bytes?

Well, yes – if we didn't have the BIOS. Along with system health-check and data loading facilities, the BIOS also includes a small set of routines for basic input and output (hence the name). We can ask the BIOS to print a letter to the screen, or check the keyboard for input, without having to write specialised drivers which could require thousands of lines of code. So the BIOS acts as a very rudimentary hardware abstraction layer, letting us do a handful of jobs quickly and easily.



See www.ctyme.com/intr/int-10.htm for a full list of routines provided by the BIOS.

2 WRITING THE BARE-METAL CODE

So, let's write some code that fits into this 512-byte space. The following is a short program that prints coloured messages on the screen for infinity – well, until you power off the computer. Type it in and save it in your home directory as **boot.asm**, or grab it online from www.linuxvoice.com/code/lv014/boot.asm.

```

BITS 16

mov ax, 07C0h    ; Where we're loaded
mov ds, ax      ; Data segment

mov ax, 9000h   ; Set up stack
mov ss, ax
mov sp, 0FFFFh ; Grows downwards!

mov ah, 0       ; Set video mode routine
mov al, 0Dh     ; 320x200x16 colours
int 10h        ; Call BIOS

loop:
mov si, text_string
call print_string
inc bl         ; Change colour
jmp loop

text_string db 'Bare metal rules!', 0

print_string:
mov ah, 0Eh    ; Print char routine
.repeat:
lodsb
cmp al, 0
je .done
int 10h        ; Call BIOS
jmp .repeat

.done:
ret

times 510-($-$$) db 0
dw 0AA55h     ; Boot signature

```

If you followed the last two assembly language tutorials, some of this will be familiar to you, but a lot of it is new as well. This is largely because we no longer have access to the Linux kernel to handle

various tasks – but we can talk to the BIOS. The first line, **BITS 16**, is a directive that tells *NASM* (the program that converts assembly language code into binary for the CPU to execute) that our code is 16-bit. When you switch on an x86 PC, it initially operates in 16-bit mode, like PCs of the early 1980s, for backwards compatibility reasons. Modern operating systems like Linux and Windows use various instructions to switch the CPU into 32-bit (or 64-bit) mode, but we don't need that here – we just want to print some text.

Now, the BIOS loads our 512-byte program into position **07C0** (hexadecimal) in RAM, which is equivalent to 1984 in decimal. (It doesn't load it into position 0, as that's taken up with some important system data.) In the first two **mov** instructions in our code, we set the data segment register (**DS**) to point to this **07C0h** location. Segments are ugly old

And here's the code running off a USB key (emulating a floppy drive) on an Asus laptop. This is the real deal.



remnants of 16-bit code, and we won't deal with them extensively here, but in a nutshell: in a 16-bit register you can store numbers from 0 to 65535. So when using 16-bit memory addresses, you can only access 65536 memory locations – that is, 64k. This is much too small for many tasks, so before 32-bit processors became the norm, 16-bit CPUs used “segments” as offsets to access more RAM.

They made 16-bit programming a mighty pain in the rear, and everyone was happy to move to 32-bit and have easy access to 4GB of RAM. Because our

“512 bytes is tiny, but cunning coders can eke quite a bit of functionality out of limited space.”

program is tiny, we don't even need to do any complicated operations with segments, and the chances are that

you will never have to again in the future – unless you want to write a 16-bit program larger than 64k.

Anyway, we then have three more **mov** instructions which set up the stack. We place the stack in a certain segment using the **SS** (stack segment) register, and then put **SP** (the stack pointer) at position **FFFFh**. If you've been brushing up on your hexadecimal knowledge since last month, you'll know that **FFFFh** = 65535 in decimal. So why are we putting the stack pointer at the very final position in a segment? If we push something onto the stack, won't it overflow and cause problems in the program?

Well, no. You see, on x86 PCs the stack grows downwards, so when we push a 16-bit (two byte) number onto it, the stack pointer is actually decremented by two bytes. When you pop a number off, it goes back up. (If you keep popping off more than you've pushed on, it will go up over 65535 and you'll have lots of fun and games in your debugging...)

Taste the rainbow

So, we've done the ugly segment-related work, and now we can get our hands dirty with some actual code that does interesting stuff. First up, we want to switch to a graphics video mode so that we can easily print coloured messages. That's what these three

lines do:

```
mov ah, 0 ; Set video mode routine
mov al, 0Dh ; 320x200x16 colours
int 10h ; Call BIOS
```

Do you remember from the previous tutorials that we called the Linux kernel using **int 80h**? Well, to access the BIOS we use **int 10h**, and the BIOS also needs various parameters supplied in registers. Normally you place the BIOS routine you want to use in the **AH** register, and then extra parameters in the other registers. For instance, to change the video mode we need to place zero in **AH** – and how do we know that? In the olden days we'd have a thick book detailing the BIOS's inner workings, but today we can find a list of BIOS routines on the web, eg www.ctyme.com/intr/int-10.htm.

You'll see there that there are routines for “Set video mode”, “Write graphics pixel”, “Teletype output” (which we'll use in a moment) and so forth. If you click on the **int 10/AH=00h** link you'll see a list of video modes underneath, and here we're using **0Dh**, which is 320x200 pixels in 16-colour mode. That's ridiculously low-res by today's standards, but ensures that the code will work almost everywhere, including on that old late 80s box gathering dust in your attic.

Then we have a loop:

```
loop:
mov si, text_string
call print_string
inc bl ; Change colour
jmp loop
```

This calls a **print_string** routine (which we define underneath). The routine takes the location of a zero-terminated string in the **SI** register, and a colour in **BL**. This loop goes on forever, and in each iteration we increment the **BL** register, so it goes from 0 to 255 and then flows over back to 0. This gives us a constant cycle of colours for the message text.

Underneath, you can see that the **print_string** routine is somewhat similar to the one we implemented last month, albeit simpler as we don't have to work out the string length. This time we use the BIOS's teletype routine, **0Eh**, which prints a character to the screen and moves the cursor

PRO TIP

It's important to note that you can't access BIOS routines from inside Linux or Windows. They're only available in 16-bit mode, and by the time that Linux, Windows and other 32-bit and 64-bit OSes have booted, the BIOS is no longer accessible. (Well, there are some stunts you can pull to set up virtual 16-bit modes, but it all gets rather hairy.) So if you want to write your own all-singing, all-dancing 32/64-bit OS, you'll have to say a tearful goodbye to the BIOS and write your own keyboard, screen and storage drivers.

Running on real hardware

If your PC happens to have an inbuilt floppy drive, you can write the virtual disk image to a real disk using this:

```
dd if=floppy.img of=/dev/fd0 bs=1024
```

You may need to do this as root, and if it's a USB floppy drive, change the device to **/dev/sdb1** or similar – use **dmesg** after plugging in the drive to see its device name. Then you can boot your PC from the floppy disk and see your code running natively on your PC.

Chances are that you haven't used floppy disks in many years, however, but there's another option: USB keys. Many BIOSes have the facility to load a floppy disk image from a USB key and execute it like a real floppy. Note that this will completely erase the USB key until you next reformat it! Plug in the key and then enter **dmesg** in a terminal. In the most recent output at the end, you'll see various messages like this:

```
sd 2:0:0:0: [sd] 501760 512-byte logical blocks
```

This tells us that the drive we plugged in has the device name **sd** – it may be different in your case. Unmount/eject the drive using your file manager (or the **umount** command at the command line), and then write the floppy drive image to the key as follows:

```
dd if=floppy.img of=/dev/sdc bs=1024
```

Be sure to get this exactly right, and replace **/dev/sdc** with whatever you saw from the **dmesg** output. Ask on our forums (<http://forums.linuxvoice.com>) if you get stuck.

Once the data has been written and you're returned to the prompt, restart your PC and in the BIOS boot menu, choose to boot from the USB key. All being well, you'll see the colourful messages again, but this time running on your very own hardware. How cool is that? The answer is: very cool.

onwards. The specific character is provided in the AL register (which we retrieve via the **lods** instruction) and the colour is set in **BL** as mentioned. So in this subroutine we keep retrieving characters from the string and printing them via the BIOS (**int 10h**) until we hit a zero, and then we **ret** (return) to the calling code.

One thing to note here is the labels with periods in front of them, eg:

.repeat:

The period denotes that this is a local label, and *NASM* extends it by prefixing it with the nearest full (non-period) label above. So *NASM* turns this into **print_string.repeat** when it works through the code. Why is this useful, you may ask? Well, it means you can use the same local label name multiple times in your code. In a big source file, you may want to use lots of labels like **loop**, **repeat** or **finish**. With local labels, each routine can have its own versions of these – you don't need to come up with unique names every single time.

The final two lines in our code aren't instructions, but directives for *NASM*:

```
times 510-($-$$) db 0
```

```
dw 0AA55h ; Boot signature
```

For the BIOS to recognise and load our program, it has to be exactly 512 bytes in size and end with the number **AA55h**. So the first line here pads out our program with zero bytes until it reaches 510 bytes in size, and then we define a "word" (a 16-bit or two-byte value) of **0AA55h** to put at the end.

And that's it! We haven't done a huge amount here, but you can add more of your own code to this bootloader, providing that the resulting binary doesn't grow any larger than 512 bytes. (If your code becomes too big, *NASM* will complain when you try to assemble it.) 512 bytes may seem tiny, but cunning coders can eke quite a bit of functionality out of this limited space, as shown in the "512-byte OS contest" at <http://forum.osdev.org/viewtopic>.

3 RUNNING THE CODE

To see our code in action, we can boot it in a PC emulator. And it also needs to be on some kind of media. The simplest way to do this is to create a virtual floppy disk – ie a disk image – so install the **dosfstools** package from your distro's repositories and enter this command:

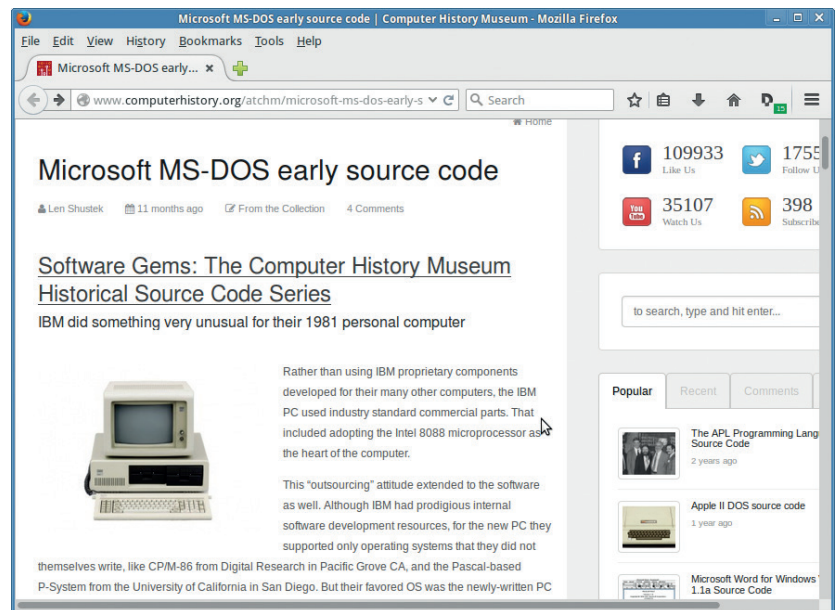
```
mkdosfs -C floppy.img 1440
```

This creates a new DOS-formatted disk image called **floppy.img** that's 1.4MB in size. Next, assemble the code:

```
nasm -f bin -o boot.bin boot.asm
```

The **-f bin** is important here, as we want a plain binary file – we don't need a complicated Linux executable with all its extra bits and bobs. This creates a 512-byte file called **boot.bin**, and we inject it into the start of the floppy disk image like so:

```
dd conv=notrunc if=boot.bin of=floppy.img
```



For more examples of 16-bit bootloader and simple operating system source code, see <http://tinyurl.com/dossource> (MS-DOS 1.1 and 2.0)

php?f=2&t=21042. Here, programmers were challenged to make something impressive in 512 bytes, and they certainly succeeded: one developer wrote a pseudo-3D car-racing screen saver, while another implemented the *Game of Life*.

Another project you may find useful, either as a source of code snippets you can nab, or just general inspiration, is *Tetranglix* at <https://github.com/Shikhin/tetranglix>. This is basically *Tetris* implemented in a bootloader – so inside 512 bytes – and while it's not much of a looker, it maintains the core gameplay elements of the timeless classic. Then there's *BootChess*, which proudly proclaims that it's the "smallest computer implementation of chess on any platform", weighing in at just 487 bytes: www.pouet.net/prod.php?which=64962.

Now install a PC emulator such as *DOSBox* or *QEMU* from your distro's repositories, and boot your virtual floppy disk in them using one of these commands:

```
dosbox floppy.img
```

```
qemu-system-i386 floppy.img
```

And *voilà*: the coloured messages zoom by, produced by the bare-metal code that you've just written. Not bad, eh? If you want to try it on real hardware, see the boxout, left – and next month we'll expand this bootloader considerably so that it can execute more programs from the disk, including programs you've written yourself. Yes, we'll turn it into a rudimentary, but functioning, operating system! 📺

Mike Saunders has written a whole OS in assembly (<http://mikeos.sf.net>) and is contemplating a Pi version.

MASTERCLASS

Keep tabs on your network traffic to see what's going through the wires you're paying for.

SEE WHAT'S HAPPENING ON YOUR NETWORK.

Looking at the data flowing across your network can be an ethereal experience. You'll be surprised at what you can see when you start snooping...

JOHN LANE

A network protocol analyser lets you peer inside your network and see what's really happening on it. Its deep packet inspection allows you to see the chunks of data, called "packets", moving across your network and view the data inside them. And, if that data isn't encrypted, you'll quickly realise how much information is readable by anyone!

We're going to look at *Wireshark*, a cross-platform desktop network analysis application that was known as *Ethereal* until 2006. There are versions for use with *GTK* and *Qt* libraries, although the latter is considered a less-mature port of the former. They use the *libpcap* packet capture library to capture network packets for inspection.

You'll most likely find a package in your distro's repository – you can choose from command-line, *GTK* or *Qt* variants. The *GTK* version on Arch Linux is **wireshark-gtk**; it's simply **wireshark** if you're Debian-based. The current stable release is 1.12.3, but some distros may lag that slightly.

You can run *Wireshark* as any unprivileged user, but it requires elevated privileges to capture packets and

this is granted to users in the **wireshark** group. Add your user to this group if you want to capture packets (you don't need to do this if you'll only be analysing previously captured packets).

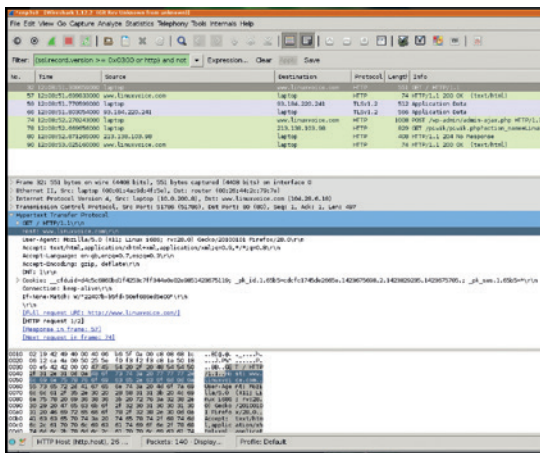
When you launch it, you're presented with the main screen. You can read about the various displays in the on-line documentation (press F1, do Help > Contents or browse to www.wireshark.org/docs to view the User's Guide – Chapter 3 describes the user interface). To begin capturing, choose your network interface from the panel on the left-hand side (it'll be called something like eth0 or, if you have *Systemd*, it might be like a more cryptic "enp3s0") and then press the green shark-fin button to begin capturing. Captured packets are displayed as they arrive; press the red-square button to stop capturing.

You can save your capture if you want to – use File > Save or use the toolbar button. As you would expect, there's an option to open a previously saved capture file. Users can work with such files without having elevated privileges, so they don't need to be members of the **wireshark** group.

LV PRO TIP

Packet-sniffing is a privileged affair. However, *Wireshark* doesn't need elevated privileges unless capturing traffic.

Wireshark's default layout displays captured packets. Click on a packet to see its contents organised by protocol. You can view data bytes in hex or binary.



Who's calling?

Among the various data displayed about a packet is its source and destination, which will either be a MAC or IP address depending on the protocol layer that the packet is from. You might prefer to see recognisable names here, especially if you're on a busy network. There are two options for you – either let *Wireshark* use name resolution or provide names yourself.

It may be tempting to use name resolution but this has the negative side-effect of polluting your capture with the name resolution requests that *Wireshark* would make to a DNS server. It's disabled by default to prevent this but you can enable it on the "Name Resolution" preferences page if you want to. However,

an alternative is to provide names yourself. You do this by creating *Wireshark*-specific “hosts” and “ethers” files to map addresses to names in a similar way to the system’s own `/etc/hosts` file and they have a similar format. Use your favourite text editor to create the file `~/wireshark/hosts` with entries like:

192.168.1.254 router

and `~/wireshark/ethers` with entries like

a2:e4:e6:65:2b:c2 router

You can also add comments – lines beginning with the `#` character are ignored.

In addition to the above, *Wireshark* will learn whatever names it can from captured DNS packets, so you might see some names resolve magically even if name resolution is disabled and you haven’t provided them yourself.

Be promiscuous

Network interfaces normally ignore any data packets that aren’t destined for them. This is usually a good thing because it removes the overhead associated with accepting irrelevant data. Data packets destined for a network interface includes the obvious “Unicast” data addressed only to it, “Multicast” data sent to multiple nodes on the network and “Broadcast” data sent to all nodes on the network.

However, data that would not normally be received can also be of interest when you are sniffing about and you can tell a network interface to accept everything that it receives by enabling its so-called “promiscuous mode”.

A promiscuous interface will see everything that arrives at that interface but that isn’t necessarily everything on the network. If it’s connected to a port on a network switch then it may only see the traffic emitted from that port. So, what you can see also depends on where you look.

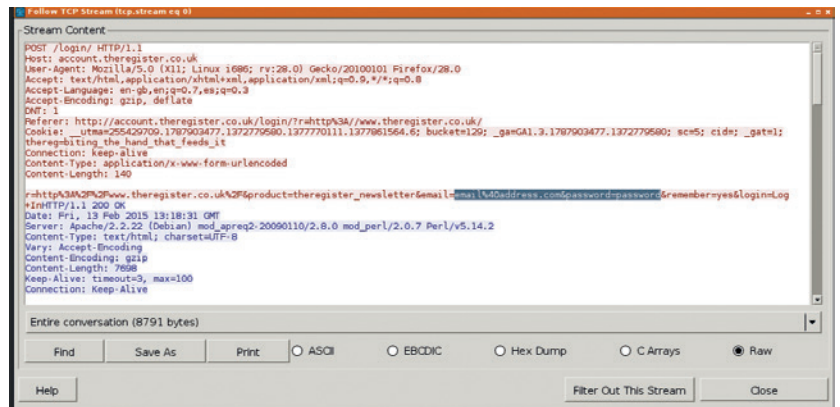
Wireshark enables promiscuous mode by default, but you can control it with a check-box on the “capture” page of the Edit > Preferences box.

Sniff, sniff...

OK, so you’ve captured some data. Let’s take a look. The first packet captured receives a time-stamp of zero and all other packets are time-stamped relative to the first. These values are displayed in the “time” field by default, but you can change this to regular date-time values by selecting View > Time Display Format from the menu.

Captures usually contain a lot of data, so the first thing worth doing is to apply some filters to make it easier to see what’s interesting. The “Filter” box is beneath the toolbar and you can enter basic expressions to filter what’s displayed (there’s also a capture filter but it’s best to capture everything while you’re learning how to use it).

To begin, let’s sniff out web browsing activity. Enter “http”, the protocol used for web traffic, into the display filter box in the toolbar and then press the Apply button (you need to use lower case in the filter even



though the display shows the protocol in upper case). The display will then show only that kind of traffic. You may also see SSDP packets because they’re based on HTTP (it’s a multicast protocol used by UPnP services). You won’t, however, see HTTPS. You can resolve these issues by using a filter expression like this:

(ssl.record.version >= 0x0300 or http) and not udp.dstport == 1900

This example demonstrates the display filtering language. It accepts the usual comparison and logical expressions written using English keywords or C-like symbols and you can use parentheses to build compound expressions. As you write your expression, a drop-down list shows possible values based on what you’ve entered. You’ll notice from our example that you can’t always use the protocol name but there are other tactics you can use like we have done to include SSL and exclude SSDP.

Another use for display filtering is to add emphasis to captured packets by displaying them in different colours according to defined “Colouring Rules”. A number of default rules are provided, but you can alter them to suit your own requirements (select View > Coloring Rules from the menu). You can import and export rules and there are user-contributed profiles available at wiki.wireshark.org/ColoringRules. The Import dialog has a button that imports the Global Colour Filter File, which contains the default rules.

Another way to apply colour filters is by conversation, or all the traffic between two endpoints – the protocol-specific places between which the traffic flows, such as the IP address and port at each end of the HTTP request made when you browse a website. You can colour the conversation that includes the selected packet by pressing the Ctrl key along with a number key.

You can right-click on a packet and select Follow TCP Stream to display the conversation it’s part of. This opens a separate window that displays the data within the entire bi-directional conversation, but you can filter this to view either direction.

There are other displays that list and categorise conversations (Statistics > Conversations) and endpoints (Statistics > Endpoints), plus many others that you can explore.

Viewing a TCP stream lets you see a conversation in one piece. Look out for those login credentials sent in cleartext...

LV PRO TIP

You can create rules based on captured packets for various firewalls including *iptables*. Right-click a packet and select Tools > Firewall ACL Rules. You can build rules that either accept or drop the selected packet.

LV PRO TIP

Filter expressions are described in the User’s guide. See section 4.13 for capture filters and 6.4 for display filters.

SNOOP WITHOUT A GUI WITH TSHARK

When you need to capture packets from a server, you need a tool that works over SSH...

JOHN LANE

Sometimes you may want to use a packet analyser on a server or other machine where it isn't practical to run a GUI application. Perhaps you want to capture packets on the server for later analysis with *Wireshark* back on your desktop, or you just want to perform some quick checks right there on the server. Command line tools exist to help in this regard and *Wireshark's* command-line cousin, *tshark*, is one of these.

It's usually packaged separately so that you can install it on servers without the GUI dependencies. Debian users should look for the **tshark** package, whereas it's **wireshark-cli** on Arch. Capturing has the same privilege requirements as the full GUI version (be root or add your user ID to the **wireshark** group).

You can run it straight from the command line and, by default, it will list the captured packets to the screen. However, it's more useful to write the capture to a file:

```
$ tshark -i eth0 -w capture.pcapng
```

Tunnel it

Another way to use **tshark** is through an SSH tunnel. This lets you monitor a remote server in real time with *Wireshark*. It's a bit convoluted, but goes something like this:

```
$ ssh myserver 'tshark -f "port !22" -F pcap -w - | wireshark -k -i -'
```

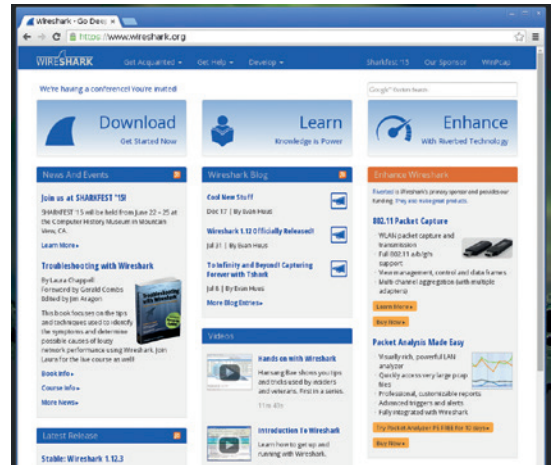
This assumes that you can log in to the server without needing to enter a passphrase (ie your SSH agent has cached it). If that isn't possible, an alternative method is to use a named pipe:

```
$ mkfifo /tmp/myserver
```

```
$ wireshark -k -i /tmp/myserver &
```

```
$ ssh myserver 'tshark -f "port !22" -F pcap -w -' > /tmp/myserver
```

The arguments tell **tshark** to ignore traffic on port 22, otherwise we'd get the SSH tunnel in our capture, and to output in **pcap** format, because that's what *Wireshark* expects on standard input. The arguments



The *Wireshark* website is a good reference for information. There's documentation and a question-and-answer system that you can use to ask for help.

given to *Wireshark* tell it to start capturing immediately from standard input.

One thing to consider is that you'll need to restart *tshark* if you stop the capture in *Wireshark*. This is obviously easier with the named pipe method because it doesn't require restarting *Wireshark* as well.

When capturing remotely, you can pass parameters on the *tshark* command-line to filter the capture and reduce the amount of data passed over the SSH tunnel. Both *tshark* and *Wireshark* delegate packet capture to a separate utility called *dumpcap*. This is the part that runs with root privileges, and you can use it directly if you don't require the post-capture features. You can replace *tshark* on the command-line like this

```
$ ssh myserver 'dumpcap -f "port !22" -P -w -' > /tmp/myserver
```

The other way to use *tshark* is to do packet analysis directly on the server. You can do this while capturing or by reading in a capture file, for which you use the **-r** argument:

```
$ tshark -r capture.pcapng
```

You can apply display filters using the **-Y** argument using same syntax as *Wireshark*. The **-f** argument we used above is for capture filters; this has no effect when reading a capture file. There's also a **-R** argument for a read filter, which discards unwanted packets read from a file.

Options by example

tshark has no interactive mode, so you need to know in advance what output you want and describe it using the command-line options. This is best described with an example.

Capture file formats

The traditional packet capture file format is called **pcap** and there are lots of applications that use it including *tcpdump* and *Wireshark*. There's also a next-generation format called **pcap-ng**, and *Wireshark*, *tshark* and *dumpcap* use this by default. It's recommended to use the **.pcapng** file extension for these to differentiate them from the **.pcap** file format.

You must use the **pcap** format if you're using pipes, because neither *Wireshark* nor *tshark* can read **pcap-ng** from a pipe. You'll get an "Unrecognized libpcap format" error message if you try to do this; it's why we specified the format in the SSH examples. *Wireshark* includes a utility, *editcap*, that can convert between these formats.

Real uses

Network sniffing might, at first, appear to be a nefarious activity, but there are legitimate reasons to do so. Here are some examples of things you might find:

- Consumers of unwanted network bandwidth.
- Malware.
- Misconfigured network devices.
- Inappropriate network traffic (hack attempts, etc).

```
$ tshark -i eth0 \
-f "tcp port 80 or tcp port 443" \
-f "host www.linuxvoice.com" \
-w capture.pcapng \
-z follow,tcp,ascii,1
```

We've broken the command across multiple lines to make it more readable. It begins with **-i** to select which interface to capture on. This argument can be repeated if you want to capture on more than one interface. You can list the available interfaces with **tshark -D**, which outputs a numbered list. You can specify interfaces by their name or number.

Next, we provide capture filters. These use a different syntax to display filters, which comes from the underlying *libpcap* library. First, we specify HTTP (port 80) and HTTPS (port 443) to see only web traffic. Then we restrict our interest to a specific host.

We then use **-w** to write out output to a file for later perusal with the *Wireshark* GUI. The last argument requests a statistics report and is equivalent to the "Follow TCP Stream" function in the GUI. The conversation is written to the terminal when we end the capture (Ctrl+C).

It's in the air

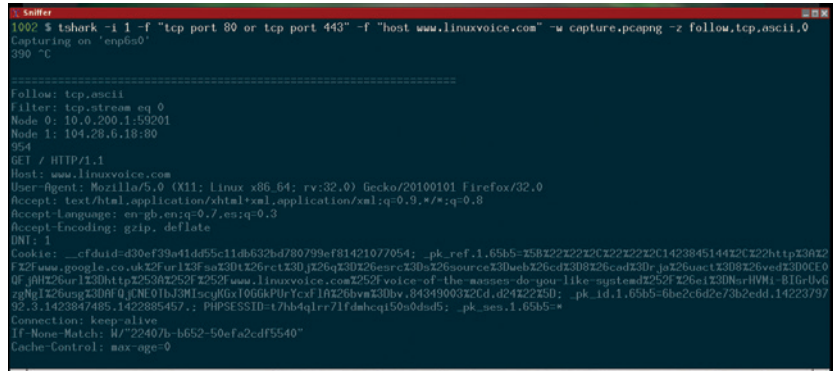
It's quite likely that the traffic you want to analyse is wireless and doing so is entirely possible but a little more difficult than sniffing copper.

If you capture packets from a wireless interface, you'll only see data after connecting to the wireless network and then only what's intended for your interface (promiscuous mode has no effect). But there's more – the wireless network protocol, called IEEE 802.11, uses its own data packets to operate the network, and you can see these as well as all the data belonging to other users of the network. You just need a wireless interface that can be put into a special "Monitor" mode.

Many interfaces don't allow monitor mode but some allow a second device node to be created called a "Radio Tap". One such interface commonly found in laptops is the Intel Pro Wireless 2200 that uses the "ipw_2200" Linux kernel driver. If you have one of these, you can enable a radio like this:

```
# echo 1 > /sys/class/net/wlp6s10/device/rtap_iface
# cat /sys/class/net/wlp6s10/device/rtap_iface
rtap0
# ip link set rtap0 up
```

You can then capture all wireless data from the radiotap device (**rtap0** in the example).



Most wireless traffic is usually encrypted, and you need to supply the passphrase to decrypt it. The **-o** command-line option allows various options to be set, and we can use this to supply these credentials. The following example demonstrates this with a sample capture file that you can download:

```
$ curl -o sample.pcap 'http://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=wpa-Induction.pcap'
$ tshark -nr sample.pcap -o wlan.enable_decryption:TRUE -o "uat:80211_keys:\wpa-pwd\", \" Induction:Coherer\" -Y "http"
```

The capture is from a wireless network called "Coherer" that has a WPA2 passphrase of "Induction". The options we set enable decryption and supply those credentials. The values given in these arguments are those you'd see if you looked at the preferences stored in the **~/wireshark** directory that *Wireshark* creates. You'll find the **wlan** values in a file called **preferences**. The **uat** (User Access Tables) values are different – you'd find those values in the specified file which, in this example, is **80211_keys**.

If there is a **preferences** directory on the machine running *tshark* then it will use it. You may find it easier to configure *Wireshark* to your requirements and then copy the **preferences** directory to where you want to run your capture. See the "IEEE 802.11" panel on the *Wireshark* preferences screen. Decryption is only possible if you capture the authentication handshake, so you need to start capturing before an endpoint authenticates with the access point.

Other tools

The library that *Wireshark* and many other tools rely on for packet capture is *libpcap*. It originated as part of a command-line packet analyser called *tcpdump* that's been the long-standing capture tool of choice on UNIX-like systems. You can use *tcpdump* instead of *tshark* or *dumppcap* for packet capture; it understands fewer protocols but is more likely to be available on a UNIX-like system without installing additional packages, so it's worth knowing about. You can pipe its output over SSH to view in *Wireshark*; follow our earlier examples but replace the *tshark* command:

```
tcpdump -U -s0 -w - "port 122" &
```

John Lane provides technical solutions to business problems. He has yet to find something that Linux can't solve.

tshark can capture packets, filter them and write to a file. When it's done, it can output details of the capture. Here we see the beginnings of a web request.

PRO TIP

Beware the permissions set on your *Wireshark* preferences files, especially those containing encryption keys, because they're world-readable by default!

/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!

The recent Mobile World Congress in Barcelona had a lot of good success stories for Linux, one of the coolest may be the “Runcible” (Edward Lear would have been proud), a sort of round not-a-phone, not-a tablet, bigger-than-a-watch. The idea behind it is apparently to challenge the conventional aesthetic of modern mobile gadgets. Well, job done! It seems to run on the Firefox OS, which is of course Linux, as it runs on a Linux kernel.

Cyanogen also had loads of new announcements. This must be one of the most exciting new open source startups for some time – growing from an online community of modders to a hugely popular alternative to Android, and now a fully-fledged company with partners choosing Cyanogen over Android in the first place. It will be fascinating to see the take-up of the Alcatel Hero 2+, which will ship with Cyanogen OS installed, and not a Google app in sight. Things like Maps, Google Play, Hangouts etc will necessarily be missing, so we might get a taste of how much of Android’s draw is the ecosystem rather than the interface.

The old stalwart WebOS was still in evidence too, not only on a selection of TVs, but also on some new, more conventional, watch prototypes.

These things are all Linux (even Android), which is good. But are they Linux enough? It is one thing having a watch that runs Linux, but if you can’t hack together a *Bash* script to pipe filtered IRC notifications to it, is it really Linux? I suggest we start a Campaign for Really Useful Linux. At the bare minimum, in addition to a Kernel, a really useful Linux OS ought to have a terminal. You know where you are with a terminal (usually somewhere in the 80s, but in a good way). Future gadget makers take note...



My Linux Setup **Selene Scriven**

Free software generalist, currently doing QA for Ubuntu.

Q What version of Linux are you using at the moment?

A Ubuntu! It’s good on the desktop, and excellent as a server.

Q And what desktop are you using? Unity, presumably...

A *Sawfish*, because it’s the only real-time programmable UI I’ve found that can handle 300+ simultaneous windows (useful if you only log out once or twice per year).

Q What was the first Linux setup you ever used?

A Slackware 96, then Red Hat, Debian, and now Ubuntu. Tried many others along the way. In nearly two decades of doing free software for a living, there have been a lot of adventures... from making last-mile mesh networks high-up on radio towers to building a working prototype of DARPA’s next deep-sea unmanned submarine, from teaching Eskimos how to make a meagre living on the internet to getting paid for breaking multi-million

dollar servers.

These days, I find myself tearing apart brand-new Ubuntu phones to find bugs and measure performance, while also trying to build the technological plumbing for community efforts both inside and outside of Canonical.

Q What Free Software/open source can’t you live without?

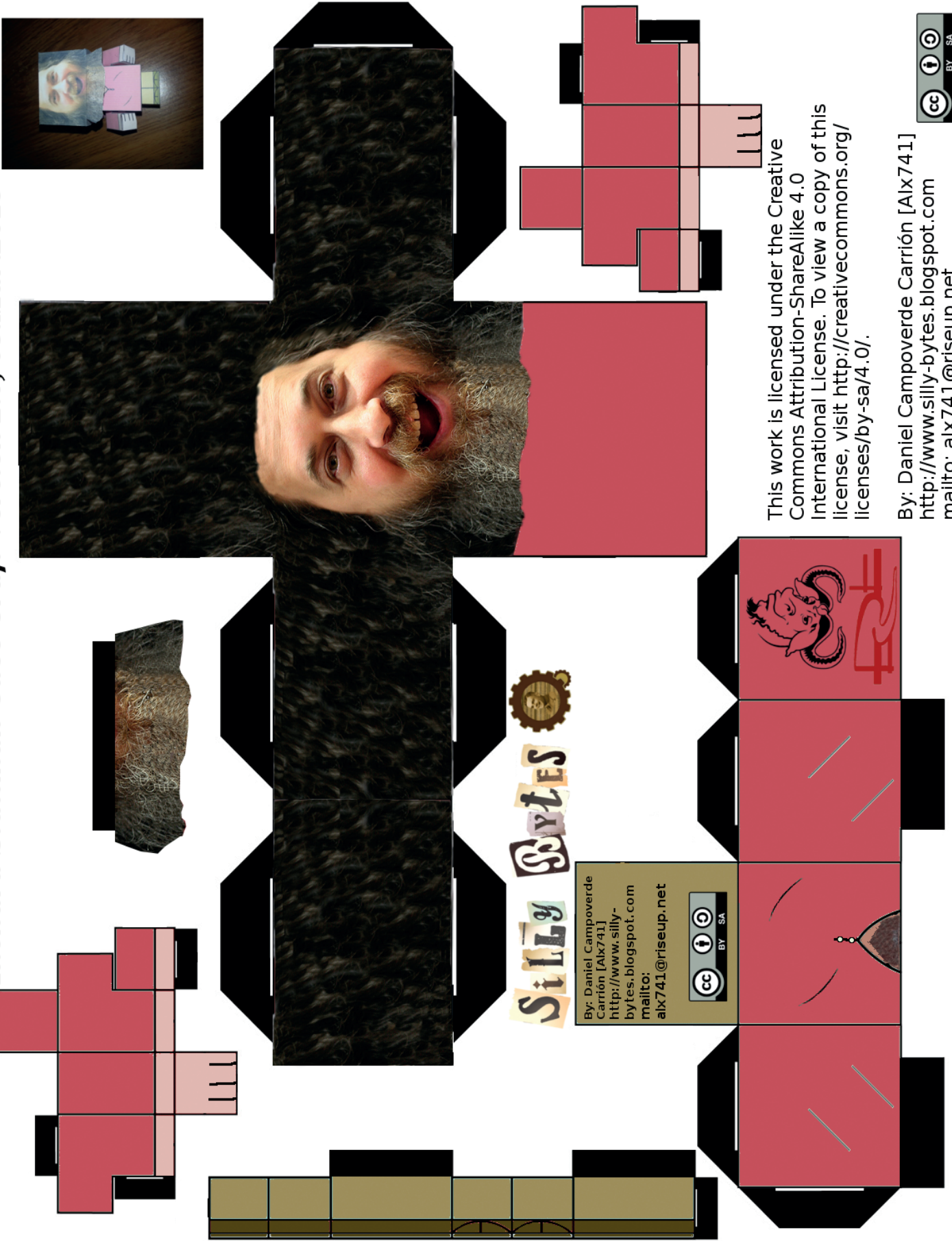
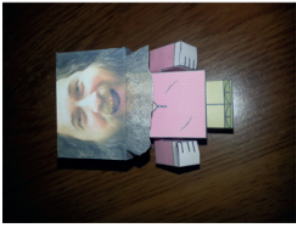
A *Urxvt*, *zsh*, *Dillo*, *Vim* and *Python*. Lightweight and powerful! They account for about 90% of my windows and 90% of my code.

Q What do other people love but you can’t get on with?

A Full desktop environments (Gnome, KDE, etc), *Systemd*, and anything else which doesn’t follow the UNIX philosophy. I get the feeling the future is going to be even more interesting than what we’ve already been through. We survived the prelude... but with free software becoming a major force in the world, the real story is just beginning. ☑

Richard Stallman Cube Craft Version 2.0, 08.Mar.2013

LINUXVOICE



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.

By: Daniel Campoverde Carrión [Alx741]
<http://www.silly-bytes.blogspot.com>
mailto: alx741@riseup.net



If you support open source software, The Open Source Initiative needs your support.

I love
open source
& want to help

I support
open source
software
and the OSI

I support the free & open source movement

Open is the better way

Open Source is an idea that has changed the world for the better
The OSI helps keep that idea alive

Wow. Been seeing a ton of tweets from @OpenSourceOrg the last few weeks
Glad to see it's getting some energy back

Because I believe in open source

I deeply believe in the OSI mission

OSI represents hope

I am a firm believer in what the OSI, and other open source communities, are doing to keep software transparent

I want to help the OSI because it is time to give back after all the support they have provided

The OSI is redefining business

I have enjoyed the benefits of open source for many years and, now an open source contributor, would like to help support the open source infrastructure

Open source shapes our future

The OSI promotes and protects

The open source movement needs this kind of support

The future depends on new ideas, alternatives and paradigms

open source

Open source software also spurs innovation and collaboration allowing different people to experiment with their own ideas about how the software could work

Open Source Initiative provides a powerful community from across the globe

The **Open Source Initiative** (OSI) was founded on February 3rd, 1998, or "2/3/98" and in 2012, the OSI transitioned to a membership-driven organization, offering both Affiliate Memberships to non-profit open source projects and Individual Memberships to open source software developers, enthusiasts, supporters and end-users.

To celebrate both our founding and promote our commitment to a representative, community driven organization, the **Open Source Initiative** is launching our first ever

Individual Membership Drive

on our anniversary — February 3, 2015 — with a goal of 2,398 new Individual Members.

We invite you to join our Affiliate Membership, including Debian, Drupal, Eclipse, FreeBSD, Linux Foundation, Mozilla, Python, Wikimedia, Wordpress and many more, as well as our corporate sponsors, Google, HP, IBM, Linux Journal and Nginx to name a few, and of course the many other Individual Members from around the globe, who have already committed to promoting and protecting open source software development, projects and the communities that ensure its continued adoption and success.

Please join now at:

www.opensource.org/join

The **Open Source Initiative** is a member-driven non-profit corporation with global scope formed to educate about and advocate for the benefits of open source development and to build bridges among different constituencies throughout the open source community. Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, greater reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. As a member of the Open Source Initiative you'll help develop and drive the direction of open source software, ensure the integrity of the Open Source Definition for the good of the community and protect open source licensing, creating a nexus of trust around which developers, users, corporations and governments can organize open source cooperation.