

FIRST LOOK: RASPBERRY PI MODEL A+



LINUX VOICE

**FREE DVD
INCLUDES ISSUE 1
& UBUNTU 14.10**

January 2015

FREE SOFTWARE | FREE SPEECH

DUAL BOOTING
GRUB 2

Work with multiple operating systems

LEGAL
PATENTS

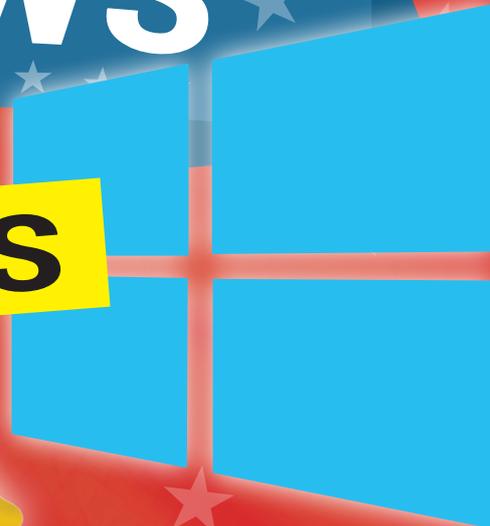
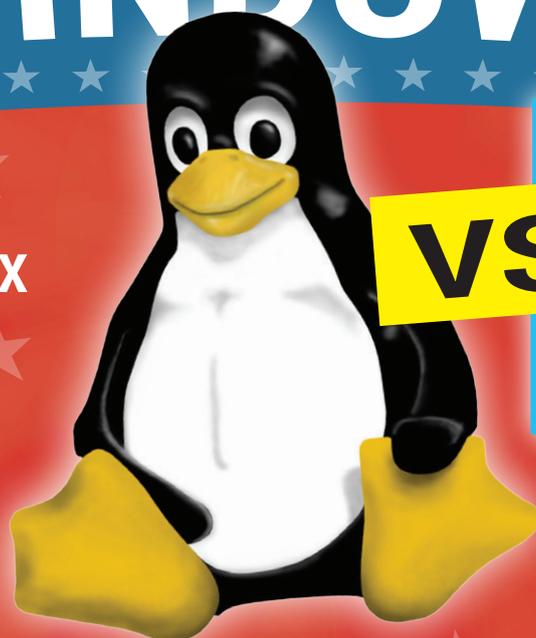
How the US courts are finally seeing sense

RASPBERRY PI
BISCUITS

Build a snack protection system with Python

LINUX THE SHOWDOWN WINDOWS

How your Linux box beats Windows 10 hands-down



34+ PAGES OF TUTORIALS

SYSTEMD Understand the changes that are coming to Linux
OGGCAMP Inside the UK's biggest hackery-makery happening
GIVE IT AWAY How we're going to share our profits with the community

GOOGLE CARDBOARD

DIY 3D

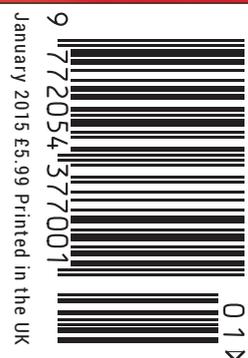
Turn your phone into an Oculus Rift



INTERVIEW

TIM BRAY

He takes your privacy seriously



9 772054 377001
January 2015 £5.99 Printed in the UK

ISSN 2054-3778
019

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

The season of goodwill

The **January** 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 relicence 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 Liam Dawe
liam@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Maligned puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Russell Barnes, Chris Brown, Mark Crutch, Marco Fioretti, Josette Garcia, Juliet Kemp, Vincent Mealing, Simon Phipps, Les Pounder, Tariq Rashid, Adam Saunders, 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.

This is our tenth issue, and as per our crowdfunding promise when we launched the magazine, it marks the release of issue number one under the terms of a Creative Commons licence (you'll find it on this issue's DVD already). The licence we chose was CC BY-SA, which means you can do pretty much anything you like with our content – even sell it, as long as you credit the source. We all feel like we're breaking new ground with this approach, and we know of no other magazine that does the same, let alone a magazine funded by the community, available on international newsstands, and run by a few people from their kitchens and bedrooms.

As our first year draws to a close, we're also starting to think about how we're going to give our profits away, and we've outlined a prototype for the process on p36. Now is the time to get involved if you want to have a say, and we'd love to have your input. Take a look at our redesigned **LinuxVoice.com** for further details, and as ever, feel free to contact any of us directly.

Graham Morrison
Editor, Linux Voice

**SUBSCRIBE
ON PAGE 62**



What's hot in LV#010



MAYANK SHARMA

My biscuits are always being stolen. Which is why Les Pounder's early warning system is so completely awesome. **p78**



BEN EVERARD

The Raspberry Pi Foundation gave us exclusive access to their new A+ model, which they've made even cheaper. **p28**



MIKE SAUNDERS

Ben combines robots, artificial intelligence and a competition in an ace tutorial about building your own Skynet. **p104**

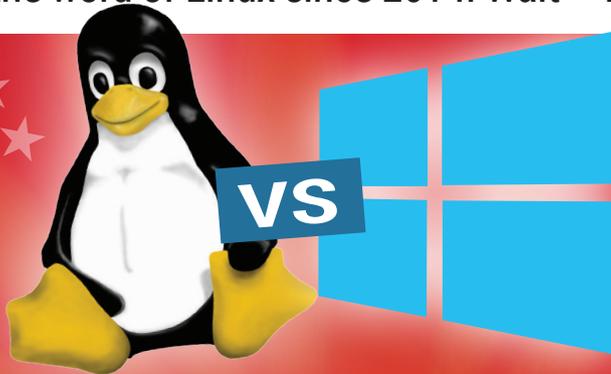


CONTENTS

January LV010

Spreading the word of Linux since 2014. Wait – it still is 2014!

**SUBSCRIBE
ON PAGE 62**



20

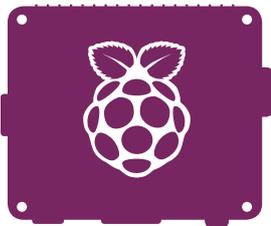
Linux vs Windows

Why we're so, so glad we're not using Windows any more.

42

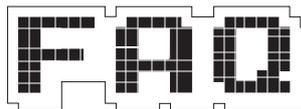
Tim Bray

The man behind the XML spec wants to talk to you about online privacy...



28 **PI MODEL A+**

Meet the latest product from the Raspberry Pi Foundation – the all new Model A+.



40 **FAQ: RISCOS**

No, this is not an OS for skydivers or fans of the Japanese delicacy, fugu. It's reduced and open.



32 **PATENTS**

Praise be for the US Supreme Court being sensible in the face of a load of patent nonsense.

REGULARS

- 06 News**
Italians won't have to pay the Windows tax any more, plus Microsoft loves us.
- 08 Distrohopper**
Netrunner, Knoppix, BackBox and Picaros Diego all come under the microscope.
- 10 Gaming**
Borderlands: the Pre-Sequel – coming to a Linux machine near you.
- 12 Speak your brains**
Another Vim convert, a silly typo and a call for freedom – all written by you lovely lot.
- 16 LV on tour**
Brought to you by Python in Dublin, OpenRISC in Munich and Open Rights in London.
- 18 OggCamp**
The biggest Geekfest in the UK extends its warm embrace to us all.
- 36 LV profit sharing**
We promised we'd give away half our profits – here's how we're going to do it.
- 56 Group test**
Six of the best lightweight Linux distros whittled down to one champion.
- 62 Subscribe!**
It's not just an operating system, it's a way of life. Join us today!
- 64 Core technologies**
Dr Brown unearths access controls: groups, users and permissions.
- 68 FOSSpicks**
Freer than a radical with a dangling covalent bond, (but not as damaging to cells).
- 110 Masterclass**
The tools to rip DVDs to video files and the knowledge to use them to do just that.
- 114 My Linux desktop**
Enter Lennart 'man behind Systemd' Poettering's Bauhaus den of geek.

TUTORIALS



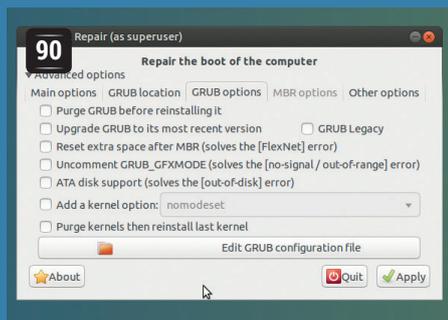
76: Learn a tiling window manager

Make better use of your screen space and kill the mouse for good.



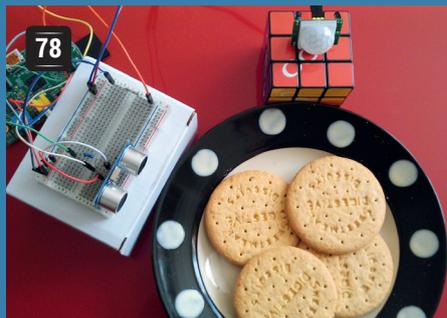
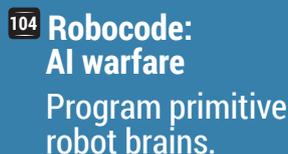
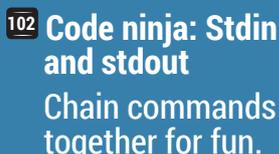
82: Markdown: Write once, publish anywhere

Publish your content in a simple format that looks here to stay.



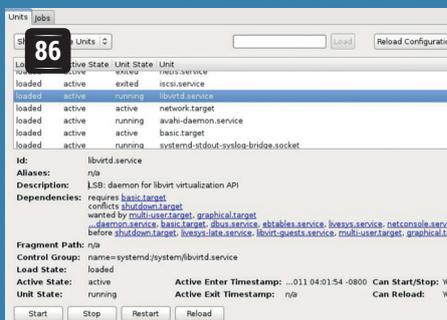
90: Grub 2: Heal your bootloader

How to fix a broken installation without losing all your data.



78: Raspberry Pi: Use different kinds of input

Use your Pi to protect a plate of biscuits from interlopers.



86: Linux 101: Get the best out of Systemd

Love it or hate it, Systemd is here – you might as well use it.



94: Olde code: Atlas – the UK's supercomputer

The transistor revolution brought supercomputing to Manchester.

REVIEWS



48 **Ubuntu 14.10**
The biggest distro pauses to draw breath before next year's big Mir/Systemd release.



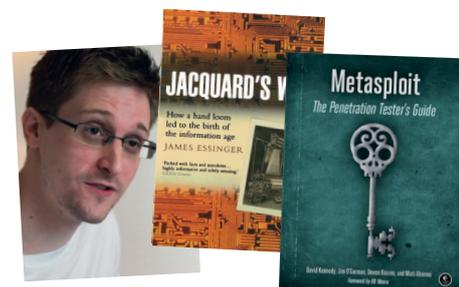
50 **Fritzbox**
If you're looking for the end of level boss of routers, we may just have found it for you.

52 **OpenBSD 5.6**
Try an alternative Unix derivative with a focus on stability and security.



53 **Google Cardboard**
Google now wants to control your senses, with its homespun 3D headset for smartphones.

54 **Books ...** and a film masquerading as a book!
Citizen Four, about the Edward Snowden leaks.



NEWS ANALYSIS

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

Opinion

The code audit mirage

What does it take to keep open source code secure?



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

In the wake of Heartbleed and other serious bugs found in open source code, I keep hearing calls for “audit”. For example, the European Commission is considering whether to budget for security audits as part of its encouragement of open source software adoption. That’s something I never heard in all my years at technology corporations. We discussed testing strategies, deployed “pen testing”, did automated code scans, used assertions in code to back up several of those test strategies, and so on. But “audit”? I never remember hearing it mentioned.

An audit is exactly the wrong strategy for open source. It spends money on finding fault without fixing anything. It reflects a worldview that open source software is a product someone else is responsible for, rather than a shared resource everyone is responsible for. If you don’t trust the code, you should be pointing your finger at whoever deployed it, not at the code. The only time to fall back to an audit is when the collaborative developers refuse to cooperate, as happened with TrueCrypt.

All code has bugs. Secret code can also have intentional back-doors and

undocumented functions. Audit is a reasonable way to give customers of this proprietary code confidence that it behaves in the way the vendor describes. But open source code is not a product until a vendor packages it as one. Until then it is a tool, waiting to be deployed by a suitably knowledgeable person.

What should we do?

First, we should be demanding of suppliers who deploy open source code as part of a deliverable with security aspects that *they* audit the code. If your supplier does not employ committers on the code they are trying to sell you, or at very least have a paid business relationship with someone who does, don’t buy.

Second, as community developers we should be performing regular scans of the code we share using a tool like *Coverity Scan*. It’s a free-of-charge proprietary offering that analyses even the largest codebase and identifies common programming errors like uninitialised pointers, buffer over-runs, unreachable code and so on. It produces a detailed, actionable report that community programmers can then cherry-pick to fix the serious issues identified. *Coverity Scan* also produces a useful code quality metric, which allows tracking of code improvement trends as well as highlighting regressions.

Third, we should keep an eye on the



One of the major reasons Heartbleed happened was a lack of new eyes coming into the OpenSSL project

diversity of communities. A project that has very few different sources of external motivation for its committers – all the same employer, for example – can easily neglect the steps necessary to keep quality high.

Open source code is neither more nor less secure intrinsically – all code has bugs, and sometimes they lead to security exposures and subsequently to exploits. But open source, properly maintained in a diverse community, can have them fixed faster and can use tools that publicly expose issues. Let’s make the code better and not allow open source to be misrepresented.

“An audit is exactly the wrong strategy – it spends money on finding fault without fixing anything.”

CATCHUP

Summarised: the biggest news stories from the last month

1 Debian fork proposed due to Systemd adoption

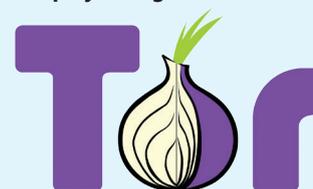
Systemd (see page 86) is being adopted by most major distros now, but not everyone is happy with the change. A group of “veteran Unix admins” have suggested that it’s time to fork Debian, after the distribution decided to use *Systemd*. The website at www.debianfork.org outlines their reasons – but it has already been parodied on www.forkfedora.org. Given the vast size and scope of the Debian project, a successful fork would be a mountain of work, but let’s see...

2 Italian Supreme Court bans “Microsoft tax”

If you buy a PC with Windows pre-installed, but never even boot that OS and install Linux instead, you won’t have much luck getting a refund for the unused software. But times are changing: the Italian Supreme Court has ruled that this “Microsoft tax” is illegal, and anyone should have the right to get their money back if they buy a PC and use a different operating system. The Free Software Foundation has a more detailed explanation here: <http://tinyurl.com/klgp5gj>

3 Tor Browser 4.0 released with Meek transports

Thanks to the new Meek pluggable transports, *Tor* can relay data via third-party servers such as content delivery networks and cloud hosting services, making it harder for governments to block *Tor* connections. www.torproject.org



4 Microsoft CEO Satya Nadella: “We love Linux”

No, this isn’t a delayed April Fool’s joke. Microsoft’s new-ish boss has proclaimed that the company now loves Linux, given that 20% of its Azure cloud platform customers are running the open source OS. This is in stark contrast to the remarks made by Nadella’s predecessor, Steve Ballmer, who famously described Linux as a “cancer”. Nadella sees things differently though: “I don’t want to fight old battles. I want to fight new ones”. A real change of heart, or just empty words?

5 Adobe discontinues Reader for Linux

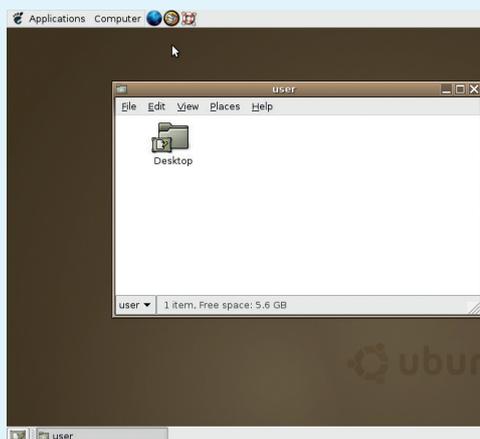
We know that plenty of Linux and FOSS users aren’t big fans of Adobe, especially given the horrendous bloatware its PDF *Reader* has become. Now Adobe has pulled the plug on *Reader* for Linux – not a big deal for many people who are using arguably superior programs like *Evince*, but frustrating for others. We asked our Twitter followers what they thought, and many of them still need Adobe’s tool for filling in certain forms which don’t work well in the FOSS equivalents.

6 ChromeOS drops ext support, puts it back in

This might not seem like a hot news story, but it shows how effective a vocal community can be. Developers working on Google’s ChromeOS platform quietly dropped support for Linux’s native ext filesystems, so you couldn’t use them on external media such as SD cards or USB keys. This led to much gnashing of teeth online, with a huge number of complaints being made, so Google relented and put the filesystem support back in. “We’ve heard you loud and clear”, said one dev.

7 Ubuntu turns 10!

Yes, the distro that introduced many millions of people to Linux and FOSS has just hit double digits. Over a decade ago, Mark Shuttleworth pored over archives of the Debian mailing lists, gathered together some of the distro’s most notable developers, and decided to get Linux onto mainstream desktops. The first release was the Warty Warthog (aka 4.10), shown on the right. Remember that chocolatey brown theme? Anyway, happy birthday Ubuntu – here’s to another 10 years!



8 W3C finalises HTML 5 standard, finally

This isn’t directly related to Linux, but it’s big news for OSes that use web apps, such as Firefox OS. After eight years of work, the World Wide Web Consortium (W3C) has finalised the HTML5 standard. Sure, many web browsers have been implementing HTML5 features for a while, such as improved media support, but now that the standard is finished, browser devs can be sure that the specs won’t change beneath their feet. <http://tinyurl.com/ohlz4x6>

DISTROHOPPER

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

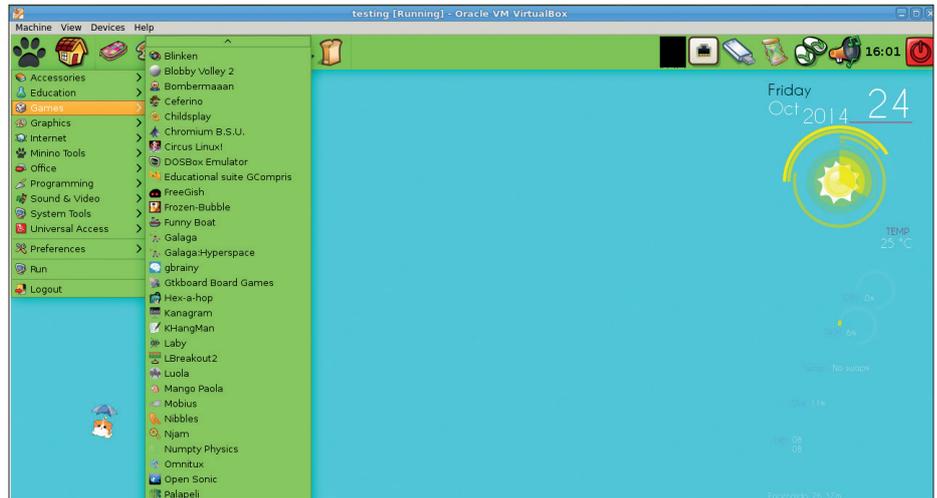
Picaros Diego

Linux for children.

There are a few distributions aimed at children: Doudou springs to mind, and there's also Sugar on a Stick. Both of these are based on the idea that you need to protect children from the complexities of the computer (and protect the computer from the children). Picaros Diego is different. There's nothing stripped-down or shielded from view. Instead, it's a normal Linux distro with a brighter, more kid-friendly interface.

The desktop wallpaper perhaps best exemplifies this. On one hand, it's a colourful cartoon image designed to interest young children. Some of the images on the landscape are icons for games, and this should encourage children to investigate the system rather than just relying on menus. On the other hand, it still displays technical details such as the CPU usage and the RAM and Swap availability.

It has kids' games and more technical software such as the *Midnight Commander*



We were too busy playing *Secret Mario* on Picaros Diego to write a witty or interesting caption.

file manager. In the programming category, we were slightly disappointed to discover it only had Gambas (a Visual Basic-like language), and not more popular teaching languages like Scratch or a Python IDE. However, it's based on Debian, so you do have the full range of software available through **apt-get**, so this isn't a problem.

The project website lists it as suitable for children aged three to 12. We think three is a

little young for a system like this, but the it may well work for children on the upper end of that age range.

Overall, we like the philosophy of wrapping Linux is a child-friendly package, but not dumbing it down. Picaros Diego won't work for every child, but if you have a budding geek in your midst, it could be the distro to bridge the gap between highly simplified toy distros and normal operating systems.

BackBox

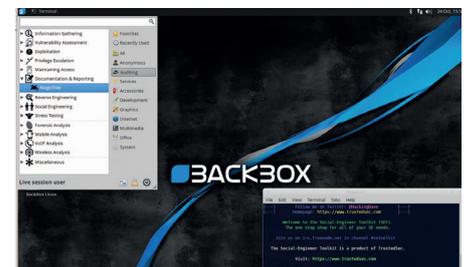
Penetration testing never looked so lovely.

BackBox is a distro for people interested in computer security (call them whatever you like: crackers, hackers, penetration testers or annoying sods who cause you to have to deal with selinux). It's based on Ubuntu with a customised LXDE graphical environment, and a whole lot of tools to help you crack, sidestep or otherwise bypass security measures. There are also additional tools for pentesters such as the *MagicTree* reporting software.

As is customary in pen testers' distros, the applications menu includes all of these tools

even though they're command line-based, and clicking on them brings up a terminal window with the help text of the tool. This might seem a little strange to people used to regular Linux distributions, but it makes it much easier to get started with the tools. Especially if – like this author – you have a terrible memory for commands you only use occasionally.

We're relative newcomers to Backbox, but it's quickly becoming our favourite penetration testing environment. It looks nice, it's easy to use, it's got the full set of Ubuntu software available (you can add



For most of the time we've known about this distro, we've been pronouncing it Blackbox. We apologise to all concerned.

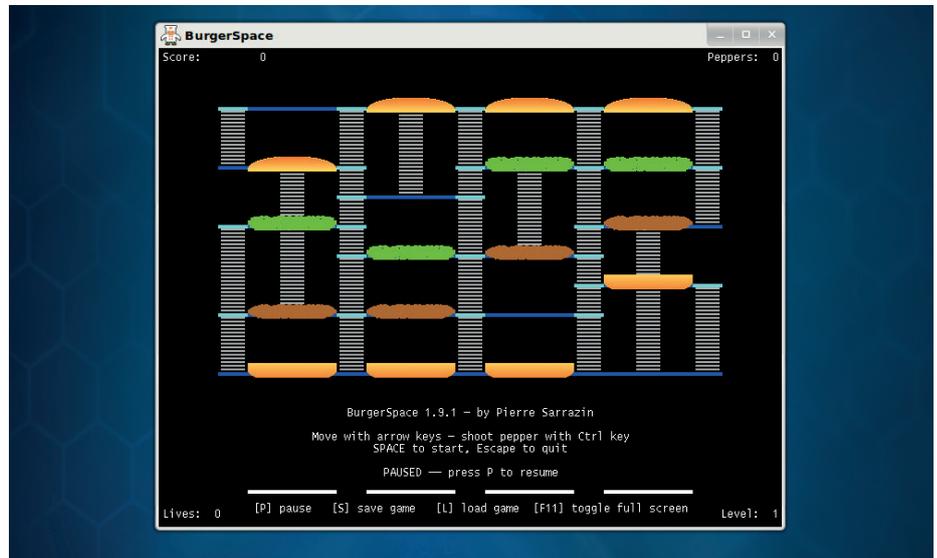
PPAs for just about anything) and it's lightweight. If you're completely new to this field, you might be better off with Kali, as there's more support for this distro, but if you're fairly familiar with Linux, you shouldn't have any trouble at all with Backbox.

Netrunner Rolling Release

A surprise find on our torrent server.

We're running a Torrent server at Linux Voice, partly to help take some of the bandwidth load off individual distro projects, but also because it provides us with a useful insight into what's popular at the moment. One of the surprise successes has been Netrunner Rolling Release. After Knoppix (see below) and Tails, the distro for the privacy-conscious, it has the highest share ratio of any of the distros we're hosting. Share ratios are an odd statistic (they provide a rough indication of the number of times something has been uploaded), and aren't an accurate representation of popularity. Still, this is a high position for a relatively unknown distro.

The main version of Netrunner is based on Kubuntu, while the rolling version is based on Manjaro (and by extension, Arch). This means that underneath, there's quite a lot of difference between the two, even though they both carry the Netrunner name. Both have a great KDE interface – not just the usual defaults, but an interface that's been crafted to look nice out the box – but below this they have little in common.



Netrunner Rolling also comes with a few games such as *BurgerSpace* to help you unwind after a stressful upgrade by smashing hamburgers.

Firefox in Netrunner comes with Adblock plus installed, and DownloadHelper (an extension to help you grab images and videos to watch later). It's little tweaks like these that make using Netrunner pleasant.

There's space for distros that package the vanilla versions of software, but we prefer it when distro maintainers put their personal stamp on the software and help the user get set up with a good system.

Knoppix The original live Linux distro is still going strong

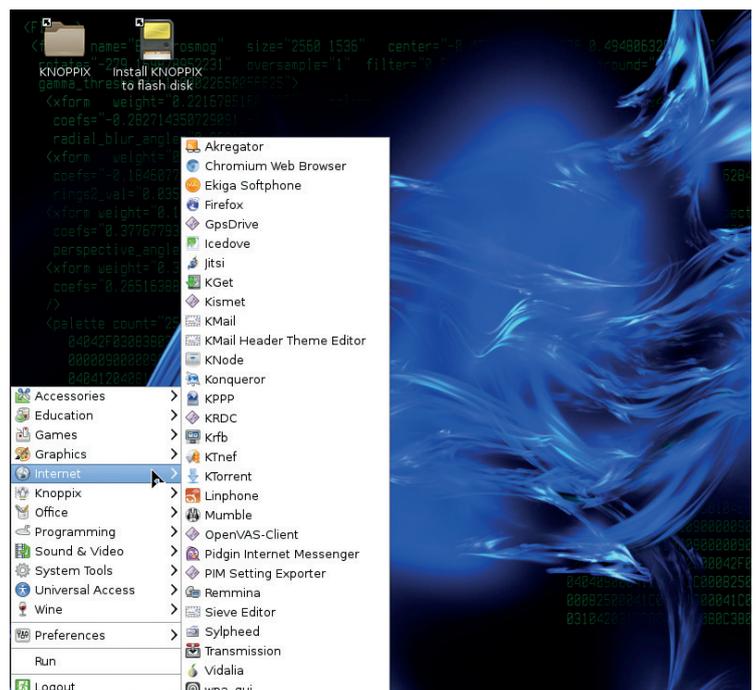
There was a time – far back in the distant past – when you couldn't try Linux without installing a distro. Then along came Knoppix, with the revolutionary concept of running Linux directly from a DVD without installing anything, and the live distro was born.

These days, of course, almost every distro comes as a live CD (or DVD) that you can try before installing directly from the live session. So is there still a place for a live-only distro like Knoppix?

Absolutely! The simple reason is that while many distros can run live, most aren't designed to do so as a matter of course. Knoppix is. This means that it comes with a wallop bundle of software that should be sufficient for almost any situation. For example, it comes with everything you should need to get online on almost any type of connection. ISDN line? Fine. 3G dongle? No problem. Need to start an SSH server? Included. In fact, it's hard to find an area that Knoppix doesn't come with software for. It can help you recover a broken machine, or operate as a main operating system in a machine with no hard drive.

With a live distro, you're also stuck with the look and feel that comes pre-configured (unless you want to make changes every time you reboot). This means it's that even more important than normal that it looks good. Knoppix has a slightly unusual combination of LXDE (usually known for its minimalism) and some pretty heavy graphical effects that we're more used to seeing on KDE. However, the end result is a nice looking desktop that should run reasonably on most hardware.

Knoppix is one of those distros that you should always have a copy of, because sometime, probably when you'll least expect it, it'll get you out of a sticky situation.



The Knoppix desktop might be a little unorthodox, but it works well.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons 

DON'T PANIC!



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

Are you feeling nervous about Steam Machines and SteamOS? Are you worried about what will happen to Linux gaming if they fail? We are here to settle those worries over a hot cuppa with some thoughts of our own.

Let's start with some facts about where Linux gaming is right now; Valve's juggernaut of a store is now serving up over 700 Linux-compatible games with no sign of slowing down. Only recently have some big AAA-class releases hit Linux too, so not only are we getting more, but we are getting games with much bigger budgets too.

GOG deserves an honourable mention now as just recently it hit 100 Linux games in its catalogue, and considering that it has only supported Linux since July of 2014 that's a pretty quick jump. GOG will also support Linux with its recently announced Steam-like client called 'Galaxy', and since it has no ties with Valve and Steam it will continue supporting Linux regardless of what happens with the Steam Machines.

Why are we noting all of this? Well, this success has all come without a single Steam Machine being released, so with all this happening for us gamers we think that Linux gaming will carry on being perfectly healthy even if Steam Machines are a flop.

A Linux-based gaming machine won't exactly be an overnight success anyway, but with Valve we should see a slow and steady push to increase Linux games to help them succeed with their Linux-based initiative.

Borderlands: The Pre-sequel

Shoot 'n' loot on the moon.

Not finding enough loot in *Borderlands 2*? Well we have you covered, as *Borderlands: The Pre-sequel* has been released on Steam for Linux.

Borderlands: The Pre-sequel is a mix of frantic first person shooting with RPG mechanics that works out rather nicely. As you shoot and loot your way through the game you will be able to level up, and unlock special abilities.

The *Borderlands* series is known for injecting a fair bit of comedy into the game, so don't go thinking this is some sort of *Call of Duty*-class serious shooter.

What makes the *Borderlands* series so different is a mixture of a few things, the most important of which is the loot system. Each gamer could end up with a completely different arsenal of weapons compared with your friends online. Are you jealous of their sniper rifle that does fire damage? No problem: just show off your assault rifle grenade launcher (and yes that's a thing!).

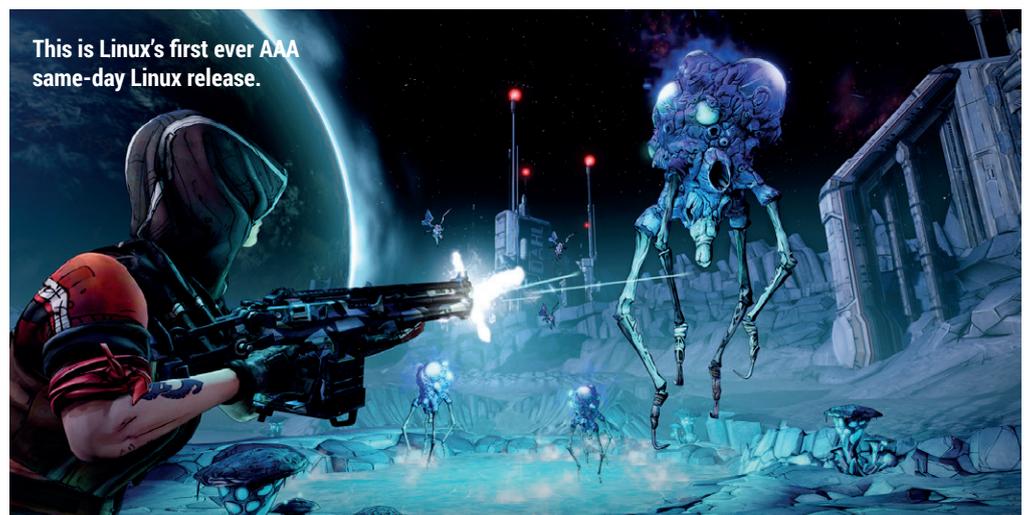
The online modes are also what makes it a great experience, as it's all completely seamless. Teaming up with people is as simple as sending



them a quick invitation on Steam, and like magic they enter your game world.

The day has come for Linux gamers to truly show our numbers. This is the first time we have been able to give a major publisher sales statistics from Linux gamers to compare them properly side-by-side with Windows & Mac, so it's time to show them our numbers!

Publisher 2K Developer Gearbox/Aspyr Media
Release Date 17 October 2014 **Store** <http://store.steampowered.com/app/261640>



This is Linux's first ever AAA same-day Linux release.

"Borderlands: The Pre-sequel is a mix of frantic first-person shooting with RPG mechanics."

Super Win The Game

Here comes the nostalgia train!

Remember the days of being frustrated trying to complete platformers on your old games consoles? Well *Super Win The Game* should bring back a lot of memories, as it seeks to emulate the look and feel of a lot of retro games, so you could call it a homage to retro games, and in our eyes it's one of the best at achieving this effect.

The game comes with full game-pad support as standard, so it can really feel the part and a game like this feels better with one too.

Super Win The Game is the sequel to the free hit indie game *You Have To Win The*

Game which is also available on Steam for Linux, and it expands on it rather a lot with a lot more content.

Unlike the free game that came before it, *Super Win The Game* is not just a standard platformer, as it blends a mix of adventure gaming along with some slick platforming moves.

There is no combat in this game, so if you're a pacifist looking for some non-violent fun, this is for you.

Publisher Minor Key Games **Developer** Minor Key Games **Release Date** 1 October 2014 **Store** <http://store.steampowered.com/app/310700/>



0 A.D.

Setting the future standard for open source games.

0 A.D. has unleashed its 17th alpha version of the open source real time strategy title that aims to be historically accurate.

0 A.D. is aimed at fans of games like Microsoft's *Age of Empires* series of games, and plays a lot like them. The major difference is that you don't upgrade through different ages like in the *AoE* series of games.

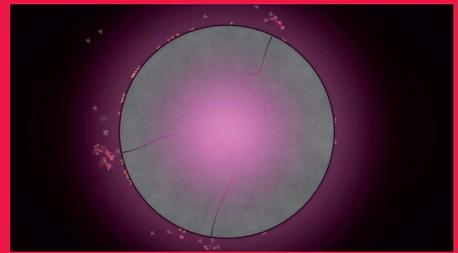
The new release is a whopper with a major rebalance to the combat, but formations have been removed for the moment to be reworked. It also includes a much smarter AI that can now deploy naval transport ships to hop to different islands, so the AI will no longer stall on naval-based maps. Combat units can now garrison walls directly for extra defence, and when you do this you can even see them on top of the walls.



New features are but a tip of the iceberg for this latest alpha as *0 A.D.* has seen a wave of performance enhancements too, so you should notice a better overall framerate, and this is probably one of the most important fixes as the late-game got rather sluggish in previous alphas.

Developer Wildfire Games (and contributors) **Release Date** No ETA for the final release **Store** <http://play0ad.com>

ALSO RELEASED...



Eufhoria HD

A revamp of a well-received ambient strategy game that is now on Linux. A great way to relax your tired mind with lovely ambient music, and easy to learn strategy controls make it a winner this issue.

The style of the game is just beautiful with the minimalistic, but gorgeous graphics. Now, go spread your seeds across the universe! <http://store.steampowered.com/app/221180>



Windward

Landlubbers need not apply. *Windward* is a procedurally generated (it's all random) naval combat and exploration game designed to be played either by yourself, or team up in co-op with friends.

You can fight against pirate ships, do a little trading or just explore the high seas. It's currently in 'Early Access', so it's not yet complete, but early reviews are positive. <http://store.steampowered.com/app/326410>



Screenshot

Do you like to play co-op with your friends and peek at their screens? We do too! Lucky for us *Screenshot* has been released in full on Steam, and the main hook for it is you actually need to use other people's screens to find them, as everyone is invisible. Madness! <http://store.steampowered.com/app/301970>

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

FREE YOUR CARPALS

This is my first letter/email ever to a magazine but I felt compelled to write it as Linux Voice is such a good read (subscription to it from my girlfriend – an amazing present).

This year, spurred on by your magazine, I have been compelled to try as much as possible to banish the mouse. I am doing my damndest to remove the rodent from my desktop to increase long term productivity. However, I am finding it very tricky.

It's tricky as there does not appear to be a clear and obvious path to doing this. Increased use of the command line (using *Yakuake* and *Terminator*), changing/learning desktop environment shortcuts and software shortcuts are the areas I am concentrating on.

I think I have got the areas I need to target but knowing which shortcuts to learn straight away and which to ignore or put lower down on the priority list is very much prone to trial and error (my desktop environment is KDE which can be baffling at times) Learning a whole slew of new shortcuts can be terminally off putting. Is there a site/book/font of knowledge that can help with prioritising which shortcuts to learn first to dramatically increase productivity. I do recommend people reduce mouse usage. Its amazing what the meta key and a cursor key can achieve together.

Laurie James, London

PS Thank you for the great *Vim* tip of always pressing the Esc key after inserting text.

PPS This email was

```
tutor3bWxd (/tmp) - VIM
6. After reading the above steps and understanding them: do it.

-----
Lesson 1 SUMMARY

1. The cursor is moved using either the arrow keys or the hjkl keys.
   h (left)      j (down)      k (up)      l (right)

2. To start Vim from the shell prompt type: vim FILENAME <ENTER>

3. To exit Vim type:      <ESC> :q! <ENTER> to trash all changes.
   OR type:              <ESC> :wq <ENTER> to save the changes.

4. To delete the character at the cursor type: x

5. To insert or append text type:
   i type inserted text <ESC> insert before the cursor
   A type appended text <ESC> append after the line

NOTE: Pressing <ESC> will place you in Normal mode or will cancel
      an unwanted and partially completed command.

Now continue with Lesson 2.
```

There's a learning curve with all-singing, all-dancing text editors like *Vim*, but once you're used to living with it, you'll never look back.

composed in *Vim* – I'm a new convert.

Mike says: Hurray! Another convert for the *Vim* cause. Welcome to our ranks, Laurie. While we do occasionally try

to go cold turkey on the mouse (RSI gets a bit much when you're constantly clicking and typing), *Terminator* is brilliant for anyone wishing to improve their productivity – if you haven't tried it yet, give it a go!

MORE TO LATEX

Of course it's important to understand the principles of markup of TeX and *LaTeX*, as Dr Sinitsyn elucidates [in LV009], but the easiest way is surely to use *LyX*, which should be available in all distro repositories. As for virus checking, it is useful to do so to avoid passing on any infected document which some Windows user might have sent to you and which you distribute further.

Clamtk is a useful graphical front end, but some distros only provide *ClamAV*, which needs use at the command line.

Dave Postles

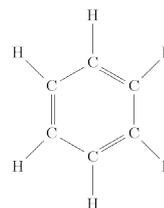
From my Trisquel Linux desktop

Andrew says: Cheers Dave – although there are no viruses out there for Linux, it's good if you're running a mail server to help keep the internet clean for poor old Windows users.

To neutralize an acid, add some alkali to get salt and water:



Benzene (C_6H_6) looks like this:



If you want most of power of *Latex* but with a shallower learning curve, give its graphical cousin *Lyx* a go.

IT'S ONLY WORDS

In your December edition you replied to Richard Bosworth about explaining free software to 'the masses'.

In a somewhat similar situation I finally worked out that the message was not so much not getting through as being totally inverted in translation. The person I was talking to turned out to treat 'free' as a synonym for 'worthless' and 'volunteer' as meaning 'someone who isn't serious'. Given that to him 'proprietary' meant 'valuable' and only a company could possibly be relied upon for support and that was it: I was snookered.

Tom Groves, Ashford, Kent

Mike says: We've had a few replies on this subject, and it's troubling that so many people are finding the



same problem. The Free = Worthless interpretation is fixed in many people's heads by years of experience, so it'll be hard to overcome. Next issue we'll go back to basics with a look at why free software matters – not just an exploration of the features (though do feel free to turn to page 20 for our thoughts on Windows vs Linux), but as an ethical choice.

The ethical stance taken by Richard Stallman and the Free Software Foundation is exactly right – it's just getting it across to the masses that's proving difficult.

AN ODD TYPO

I looked at your Issue 8 DVD and the envelope for it and was surprised to see a typo in what seemed an unlikely place – the legal disclaimer. "Legal: Linux Voice cannot accept s for any disruption, damage and/or loss to your data or computer that may occur while using this DVD." Granted, this is the typical legalistic covering of your posterior. Your readers most likely understand that and would never consider giving you any S. I can only guess what other letters of the alphabet you wouldn't want to accept.

Roy Birk

Andrew says: Hmm. It's true, we do not accept any S. We prefer pounds, euros or dollars. Or bottles of whisky. I'm as baffled by that as anyone.

PRAISE. KIND OF

I love your magazine and podcast, keep up the great work!

However in reading issue #6, I found something that my pedantic nature could not reconcile. On page 9 when discussing Red Hat Enterprise Linux 7, the article ends with "The CentOS team are working on the community build, which may even be available by the time you read this."

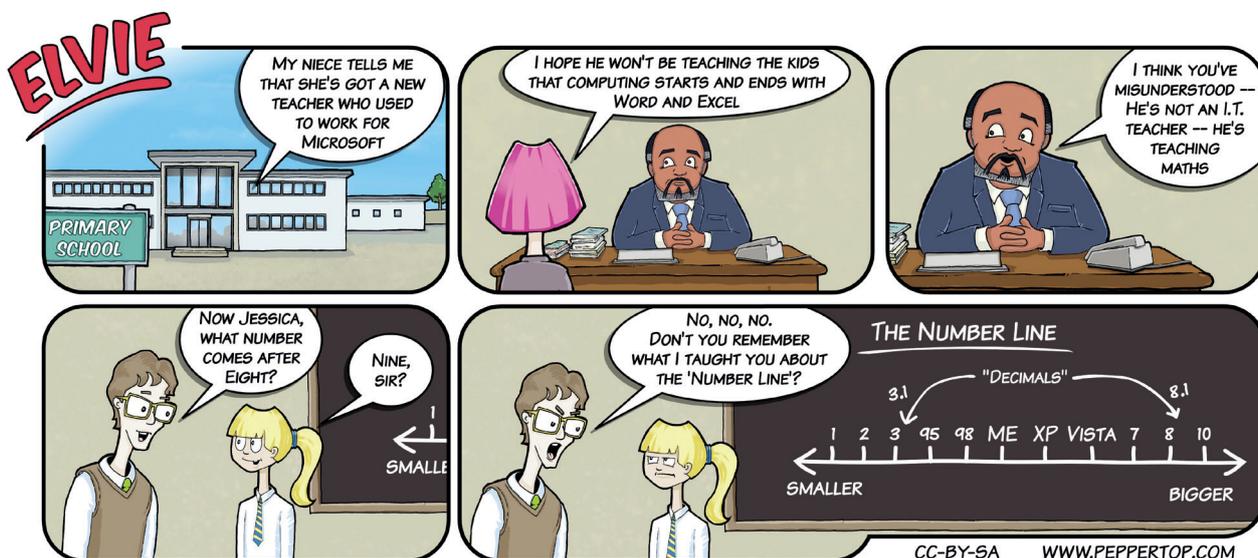
If you move on to page 46, there is a review of CentOS 7. Obviously, it was available in time for a review to make it to print. Thanks again for the hard work and dedication to the open source community.

Todd Campbell

Graham says: Rations have been reduced and the team will be flogged until performance improves.



Rest assured that we won't be accepting any S relating to this issue's DVD either.



CC-BY-SA WWW.PEPPERTOP.COM

MORE TINKERING!

I would first like to congratulate your entire team for a job well done. I got interested in Linux Voice the moment I saw your second issue, and have since become a subscriber. Just the core technology section makes the magazine a must read. I am a hardcore Windows user and though I use Linux at work I wasn't particularly attracted to it until I started reading your magazine. LV has encouraged me to try out new distros in the past few months and after trying CrunchBang, Bodhi, Manjaro, CentOS and Arch I have finally settled down with Gentoo. Formatted my Windows box and put in Gentoo! And loving every moment of it. Thanks a ton LV!

I would like to request that you include an article type for tinkerers like myself in every issue which could be about say writing your own Linux device driver (something a little more in-depth than issue 2 or perhaps a series) or tinkering with Awesome WM's configuration or just some



hardware diagnostics utilities etc.
Saptarshi

Graham says: Wow! Welcome! I know a few hardcore Windows users and none would give up their OSs without a fight, so it's wonderful to see you've found your way over to

Linux and are loving it, despite the culture shock. We'd really like to delve deeper into some subjects/projects, as you suggest, and we've got some great plans for longer form projects that spend more time relishing in the details. If anyone has a great idea for an in-depth project, let us know.

There's more tinkery fun in the pipeline – like the driver we wrote for this USB toy car.




YOUR AD HERE



Email andrew@linuxvoice.com to advertise here

WHAT DISTRO?

I started using Linux with Slackware 96, or was it 3.0, and went through Red Hat and Gentoo but failed with Arch Linux. Now I'm running Linux Mint 17 – with KDE, of course.

My requirements are simple, the internet, some games, email and multimedia. I do have a *PostgreSQL* database using *Tcl* and *Tk*.

With progressive upgrading and updating I have not gained in functionality, but have lost things. I can't get *Compiz* to do the impressive rotating cubes anymore, *giver* has been and gone, and now *World Of Goo* (I was up to chapter 2). I get segment faults, which I presume means incompatible libraries.

Where can I find a distribution, which is stable and "old fashioned", to get these back? I can understand security and UEFI

need serious attention, but a basic distribution, without updates, would give me, at least, some relief.

Ron Petch, East Gosford NSW

Andrew says: Clem Lefebvre, the chief boss of Mint, says "As a general rule... unless you need to, or unless you really want to, there's no reason for you to upgrade", which is probably a good sign that you shouldn't. We've started to fall out of love with Arch recently due to some breakages, so I wouldn't recommend that. How about Debian? You get the security updates, but there's no risk that the developers will rush forward a feature causing your software to break.

Slow and steady wins the race, if the race is all about not breaking things.

WE HAPPY FEW

I first became aware of and interested in Linux some years ago through *Micro Mart* magazine, which I read for three or four years. However, gradually there was less to interest me in it than hitherto. Then I spotted a Linux magazine, which must remain unidentified, and I started to take it on a regular basis (despite the wife's tut-tutting about the cost!). However more recently I have found that more of the content has been 'over my head' (I'm not in the first flush of youth) and I had begun to wonder whether I ought to overcome my innate product loyalty thing and discontinue.

Last week in WH Smith's I noticed Linux Voice 008, but didn't pick it up. Until, that is, in the supermarket on Friday, when I saw it again and decided to have a look. And there, to my amazement was a familiar face, Graham Morrison, no less, and then three more familiar faces hirsute editions of Andrew Gregory, and Mike Saunders, with Ben Everard

sandwiched between them. (Come back Jonathan Roberts, all is forgiven!) And there among the contributors another familiar name, Mayank Sharma. What I have read so far I have understood. I like the style and appearance of things so I fancy you have just gained a regular reader.

I have an issue you may well be able to help me with. I have for the last two years been intending to install a Linux distro on my machine. At the moment I use Windows (excuse the language!) XP and have absolutely no intention at all of upgrading. However, not being overly knowledgeable about computers and computing, I have yet to take the plunge. I wondered if there might be an experienced Linux user in the Bolton area, which is where I live, upon whose guidance and help I could call. If there is someone they can email me at AGeoffMort@tiscali.co.uk.

Finally, I noticed the letter in Linux Voice by David Walker. It

struck me as more than a little odd to expect a magazine devoted to Linux to be expected to promote something which, I quote, "has nothing to do with Linux." I was not all that surprised that you have ignored it.

A Geoffrey Mort

Andrew says: If you're looking for a version of Linux to choose, you could do a lot worse than try Ubuntu from this issue's DVD. It's low on system requirements and we've included it in its 32-bit version, which, if your machine is still running XP, I'm guessing will probably fit your hardware better than the KDE version.

And if you're looking for help with Linux, the north west of England is a fantastic place to be. Preston and Blackpool have active hacker communities, and the Manchester Raspberry Jam (the Raspberry Pi is just a Linux box, remember) is one of the biggest around – not sure about Bolton itself, but I'm sure there are people near you to offer a helping hand. And I'm glad you've found us! 



debian

LUGS ON TOUR

ORCONF 2014

Mike Saunders finally becomes an alpha geek by hanging out with CPU designers.

If you've not heard of OpenRISC before, turn to p40 for our introduction to this open source processor design. OpenRISC has been around for well over a decade, but in recent years it has seen a surge in popularity – partly thanks to the overall increased interest in open source, but also due to the boom in mobile devices. So for the last three years, OpenRISC developers, users and curious onlookers have gotten together for their annual conference, and this year it took place on the 11–12 October in Munich.

Why Munich? Well, OpenRISC is used extensively in the city's Technical University for research into processor designs. Stefan Wallantowitz, a research assistant at the university, gave a talk about OpTiMSoC, the "Open Tiled Manycore System-on-Chip" framework (www.optimsoc.org). This is a pretty complicated topic, but essentially it lets you work with building blocks, such as CPU cores, memory chips, I/O modules and other bits 'n' bobs, and plug them together to create system-on-chip designs. Like OpenRISC, OpTiMSoC is primarily created in the Verilog hardware language, though it's not intended for production purposes.

Open hardware

Meanwhile, Olof Kindgren and Stefan Kristiansson gave updates on the status of the OpenRISC project, specifically the OR1K core; they talked about plans for the future, such as a new execution pipeline and improved 64-bit chips. Simon Cook from Embecosm, an embedded systems company that sponsored the conference, gave an



Most of the talks were very technical, but there was plenty of opportunity to ask questions and learn more.

update on toolchain support for OpenRISC – that is, the current state of GCC, Binutils and LLVM/Clang. And Sebastian Macke demonstrated his fascinating JavaScript emulator for the chip, which lets you boot a fully working Linux-on-OpenRISC installation inside your web browser.

Various other talks and presentations were given (see <http://tum-lis.github.io/orconf2014> for the full list), and although it was a fairly small event with around 40 people turning up, it was great to see so much passion and enthusiasm around open

hardware. Indeed, one attendee grumbled that the breaks between presentations were too short, because everyone was enjoying meeting other people and having one-on-one discussions! Of course, the free beer during the day helped in that respect too.

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

PyCon Ireland 2014

Josette Garcia is surprised not to see Google sponsoring its local Python event...

PPyCon Ireland gathered 300 of Ireland's best and brightest Pythonistas together to share knowledge, develop their skills, discuss with peers and hopefully have some fun. We should not forget the people who came from further lands – Belgium, Hungary, France, Germany, Poland and Turkey to name but a few.

For €70, you get two days of talks, breakfast and lunch – corporate tickets are a little more expensive but students pay a lot less. As we all know, Ireland offers a lot of opportunities to the big high-tech companies such as Amazon, Apple, Cisco, Dropbox, eBay, Facebook, PayPal, Twitter and many more who in turn offer lots of jobs and helped the Irish economy to grow by 3.5% this year. I had a long chat with the guys from Brightwater Recruitment who said that the demand for techies was very difficult to fulfil – too many posts, not enough people. They are

very happy to hire qualified foreigners.

The first talk was “The Real Unsolved Problems in Data Science”, given by Ian Ozsvald. According to Ian, data science was described as “the sexiest job of the 21st century” by the Harvard Business Review. All that I heard was, “Clean your data, clean your data” – nothing new or 21st century here and definitely not sexy. What is new, though, is how Ian highlights the gritty problems and proposes ways for Pythonistas to tackle and solve these issues to keep Python as the go-to language for practical data science work. Ian wrote about his talk on <http://ianozsvald.com> where you can also find his slides.

One of my favourite talks was given by Ben Nuttall – “Pioneering the Future of Computing Education”. It is nice to see the involvement of the Raspberry Pi Foundation, which is leading the way in the computing in schools revolution by providing affordable



PyCon Ireland started with a modest 70 delegates in 2010, but by 2014 has grown to over 350 attendees.

hardware to people of all levels of experience. With their education team they are creating resources and training teachers who in turn will teach kids to start programming at a tender age.

With over 650 members, Python Ireland has had a really successful year. The monthly meetups have been very well attended with generally no less than 50 attendees turning up on the night. Talks are given on a variety of topics from computer science, finance, big data and more, with a focus on Python-related technologies. Preparations for the 6th PyCon Ireland start next month. Hope I meet you there! 📺

Open Rights Group hack day

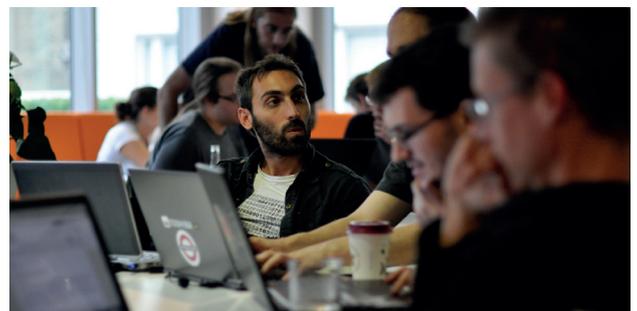
James Baster reports from a group using data for the power of good.

Open Rights Group (ORG) is a UK-based organisation that campaigns on digital issues. Recently they held a hackathon in London to work on technical projects, and a small but keen crowd gathered in Mozilla's donated office (Thanks!).

Mobile phone operators and ISPs now offer filters, which block sites that may be harmful to children. The problem is that no-one knows which sites are filtered and the filters aren't perfect and often block perfectly harmless content. So ORG recently launched www.blocked.org.uk, which was created by volunteers. This project places machines on connections so it can test and publicise which websites

are erroneously blocked. There was a nice moment at the start of the hackathon where volunteers who had been working virtually for months finally got to meet in person! This project is of course open source.

Other projects worked on included a “Kickstarter for elections”, templates to help people make requests for their personal data, and tools for making people aware when they are in areas covered by CCTV or other tracking. A calendar of campaign events is being tested, and a game specially designed to teach players about digital issues was shown – your hacker daughter has been taken and your character has to



understand issues like tracking and encryption to get her back.

Open Rights Group needs our support to continue its work – check out <https://www.openrightsgroup.org>, visit your local meetup or join the volunteers email list and get involved before the next hack day! 📺

The Open Rights Group used the hack day to put some of its ideas into action.

Photo credit: Jim Killock (CC BY-SA)



As well as spending time with some top-class geeks, we managed to sell a few T-shirts and mugs – which you can now buy at shop.linuxvoice.com.



OGGCAMP 2014

Probably the largest event of its kind in the whole entire United Kingdom.

First, a little disclosure; we, along with Canonical and the OggCamp community, were Gold Sponsors at this year's event – you may have seen the adverts we ran on OggCamp's behalf in earlier issues. We did our best to promote what was happening and all of us here at Linux Voice attended, running a stall in the exhibitors' area, meeting lots of incredible people and drinking slightly too much beer. The sponsorship also gave us the chance to record a live podcast on the Sunday afternoon (you can listen on LinuxVoice.com).

We love what the OggCamp team does and why they do it. Everyone who helps is a volunteer and they all work incredibly hard putting the event together; from the original planning, the merchandise, dealing with registration and payments, the room re-organising, staging, audio engineering, lighting,

evening events, venue and accommodation, and the countless other tasks and processes we don't know about. Without these incredibly dedicated people, the event simply wouldn't happen. So firstly, a huge thank you to them.

OggCamp is an un-conference that's free to attend, but contributions are gratefully received and genuinely help make the event a success. This year it was held at The Oxford Hotel, some distance from the famous city centre. The conference takes place over a Saturday and a Sunday, but a great number of people arrived on the Friday evening and converged at the local public house, The Plough Inn. The landlords looked terrified at the hordes of monochrome T-shirts but dealt patiently with the introverted beer orders. It was a beautiful evening with lots of friendly faces that could have lasted longer had we not drunk the pub dry

and forced them to close early. Which was definitely a good thing, as we all needed to get to the conference early for a day of talks, and talking.

Bring and buy

Our table in the exhibitors' room was next to Mark and Vince from Peppertop Designs (www.peppertop.com), the immensely talented duo behind our own Elvie comic strip. They create all their illustrations using open source software, releasing all of their assets under a Creative Commons licence, and this was the first open source/un-conference they'd been to. We have them to thank for the pen and the paper we needed to write prices on our mugs and T-shirts, as well as the general advice we needed at an event like this.

On our left were the equally wonderful Steph and Stacey from Ragworm (<http://ragworm.eu>), printing up their own circuit boards and assorted electronics paraphernalia. We also met with Swindon Hackerspace. They'd brought a long an ingenious Twitter to Teletype machine, whose tapings sounded exactly like the 1980s sports results teleprinter on BBC's Grandstand. This relic must have something to do with Swindon Hackspace being located in Swindon's Museum of Computing, and we promised to pay them a visit after the current deadline receded and find out for sure.

There were several other hacker spaces in attendance, including Oxford Hackspace, the Surrey and Hampshire hackspace and Reading's MakerSpace (RLab). There was also a fantastic soldering workshop hosted by OpenTRV, whose founder, Damon Hart-Davis, gave a talk at the event. OpenTRV is an incredibly ambitious project that's attempting to cut the UK's entire carbon footprint by 10% with a rather neat and geeky solution. In the UK, many of us have hot water radiators and yet they're normally all controlled by a single thermostat positioned somewhere in the house. This governs whether the whole system is on or off regardless of whether you use a room or not, or whether the temperature in other rooms is below the heating threshold. OpenTRV is a project based on the installation of thermostatic regulating valves on your radiators. These talk to a server, which allows you to manage when, where and how the heating is applied to your house.

We also learnt about a great idea called 'Repair Cafe Reading'. This is where local experts help

you fix your own stuff, including bikes, tools, computers, electronics, clothes and mechanical things. We thought this sounded like a fantastic project, not just because it gives equipment another life, but because it could teach a whole new set of skills. We saw the HyPi – a hydrogen powered Raspberry Pi – powering a HDMI Pi display, with a

Just some of the blinkenlights brought by Reading's MakerSpace (RLab).



As OggCamp is an unconference, the sessions themselves are both proposed and voted on in real time.



Do you have a PCB design you want made flesh? Give Ragworm a bell. They also do learner kits aimed at getting electronics hardware teaching into schools.



theoretical run time of 1.5 hours, and spent a long while talking with the various volunteers representing Hacker Public Radio.

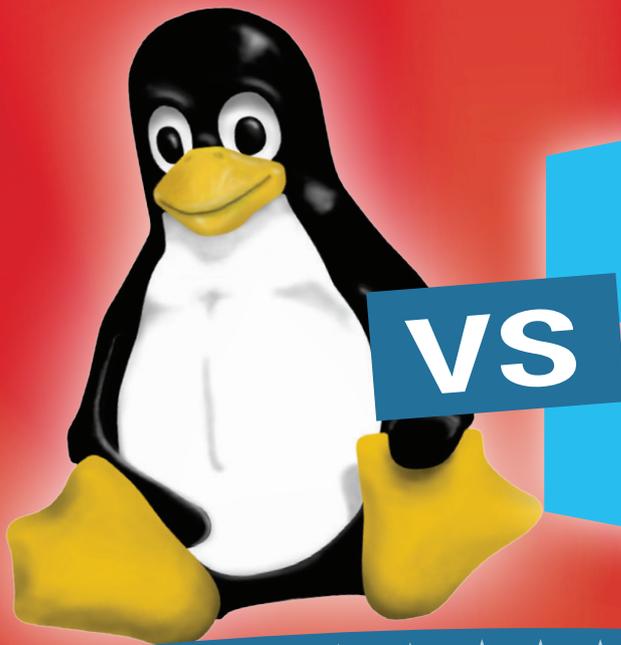
The high-tech online system for determining talks suffered a few hiccoughs, leaving attendees to the quaint but far more colourful method of "Post-It Notes on a Pull Down Metal Awning". As you'd expect, there were many great presentations over the two days; Stuart Langridge talked about cross-platform app development, Alan Pope talked about the 'Ubuntu Phone: the story so far', Andrew Katz about open hardware, and while it's difficult to believe we missed

it, there was a reportedly fantastic talk on 'Hop Hackers' and open source beer. We also have to give special credit to Steve Engledow who actually camped. At

"OggCamp overwhelmingly delivered in positivity, hackery and sheer attendee friendliness."

OggCamp. There was also an excellent podcast panel and a raffle to round the whole weekend out, leaving us all happily exhausted. OggCamp overwhelmingly delivered in positivity, hackery and sheer attendee friendliness. We just hope that the awesome organisers have recovered enough from the weekend to do the same thing next year. See you then! 🍻

HUGE THANKS TO TONY HUGHES FOR THE PHOTOS BY 2.0



LINUX VS WINDOWS

It's not enough to just be better – you have to know why you're better.

If you're reading this, then the chances are that you're already pretty fond of Linux. However, it's always good to stop and look around every once in a while to take stock of the situation, and ask yourself: is Linux still the best choice for me? We think that the answer's a resounding yes, and this month we're reminding ourselves just why we use Linux.

The more we looked at it, the more we realised that there wasn't just one reason we use Linux. It's better in loads of different ways – too many to cover fully in one article, but we've done our best.

“The more we looked at it, the more we realised that there wasn't just one reason we use Linux.”

The list looks very different today than it would have done a couple of years ago. Some things are just as great as they've always been (like package management and the shell interface), but others have got a lot better recently.

Obviously, everyone will rate each area differently. For some of you, freedom will be the most important reason; for others, it'll be security or flexibility. If you're a gamer, programmer, office worker or sysadmin, you'll have different priorities, but we've looked at areas that we think are important to everyone. Read on and remember exactly why Linux is the best OS around.

Online

The internet is a scary place... unless you're a Linux user.

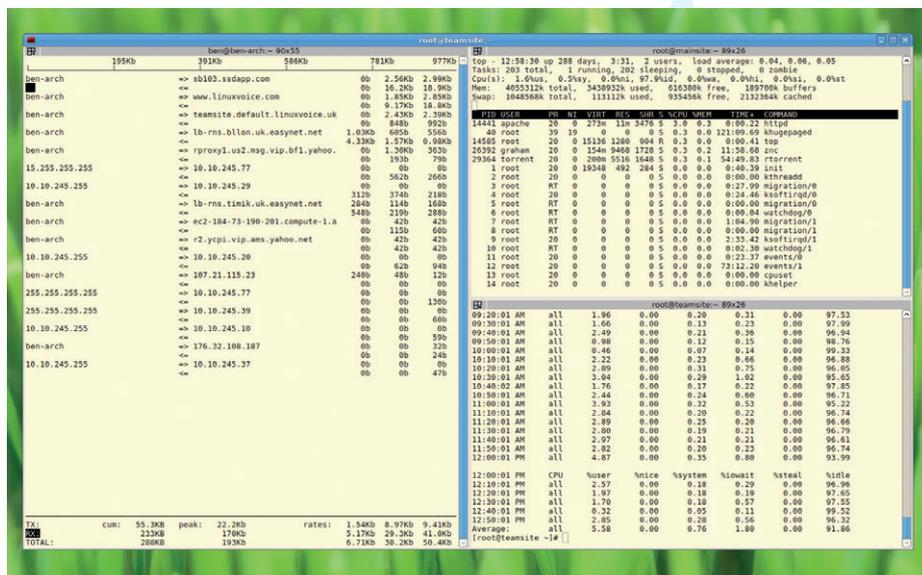
Let's be honest, the internet is built on Linux and open source. Even if you happen to be using Windows and *Internet Explorer*, if you're browsing the web, then really, you're using Linux. The majority of the servers generating the content are running Linux, and your Windows machine is just an output device.

It isn't a coincidence that Linux runs the back-end of most of the internet. The modern web wasn't created by big companies, but by little companies that dreamed big. At least two of the biggest companies in the world today (Google and Facebook) were started in college dorms less than two decades ago. Small companies like these need flexibility and cost effectiveness, and that just doesn't come from large companies with complex software licences.

The bureaucracy of proprietary solutions means that they just can't keep up with the level of innovation in Linux. Whether it's featureful filesystems, advanced virtualisation or containers, open source OSes have led the way since the web became popular. This innovation has allowed companies building their servers on Linux far more flexibility in how they deploy, and in the internet age, flexibility in your IT is a commercial advantage.

Client side

Now it's true that there are a select few things that you can only do online if you happen to be using a closed source desktop – Adobe's Air and Active X spring to mind – but these are getting fewer in number each year. Recently, the Pipelight plugin has enabled Linux users to interact with sites



Three shells running on three different computers spread out across the UK. Beat that, Windows.

requiring Microsoft's Silverlight. With the advent of HTML 5 though, more and more sites are pushing rich content through open standards rather than through proprietary add-ons.

It's not just about what you can access, there's also the issue of what can access you. The web is a dangerous place, or so some people tell say. The truth is that we've been using Linux for so long that we've forgotten all about internet-based malware. Drive-by downloads, infected adverts and others attacks simply don't bother us, not because we use antivirus software, but because we use Linux. So far, at least this operating system has proved to be far more resilient against attacks than Windows.

There have been a few scares around internet security on Linux recently

(Heartbleed and Shellshock spring to mind), but these were patched quickly, and neither one has had a significant effect on web users, since they primarily targeted servers. We shouldn't worry too much.

More than the web

The internet isn't all about browsing the web though. There are myriad methods for two computers to communicate, including FTP, SSH, SCP and RSync. Almost all of these protocols interact seamlessly with Linux.

Of course, most of these protocols work with Windows too, but they're often wrapped in ineffective GUIs, or you have to download them from a website without being able to verify their origins (see next page). Only on Linux and other Unix systems do you have full access to a wide range of power users network tools out of the box.

There is no perfect OS for the internet. Linux is more powerful and more secure, while there are still a few sites that require Windows or OS X to run. However, when everything is balanced out, we're far happier running on Linux.

- TOP 3 REASONS LINUX IS BETTER FOR USE ONLINE
- 1 Servers
- 2 Security
- 3 Flexibility



Tails

Some of the biggest benefits of Linux come from its flexibility, and few things show this as well as Tails, the live distro designed to help people securely and anonymously access the internet. It's free as in zero cost. While this is always nice, it's actually an important aspect here. If tails were built on some commercial software (like Windows), it'd be a security hole, as a payment almost always creates a record. Especially if it's done over the internet using a credit card (Bitcoin could help alleviate this, but it's still not widely used). This would enable credit card companies to track who had access to the anonymising software, and this could be used to de-anonymise it. The zero-cost access also means that it can be shared freely, copied and handed out without any restrictions. Of course, Tails isn't only great because it's free, it's also great because it delivers what it promises (a secure anonymous operating system). While it's not impossible to do that with commercial OSes, it is quite hard to do. The internals are locked off to the extent that even making a live bootable CD is difficult, let alone customising it so as to make it anonymous. Even if they were no backdoors or spyware compromising your system you could never verify it – which is why free software wins.

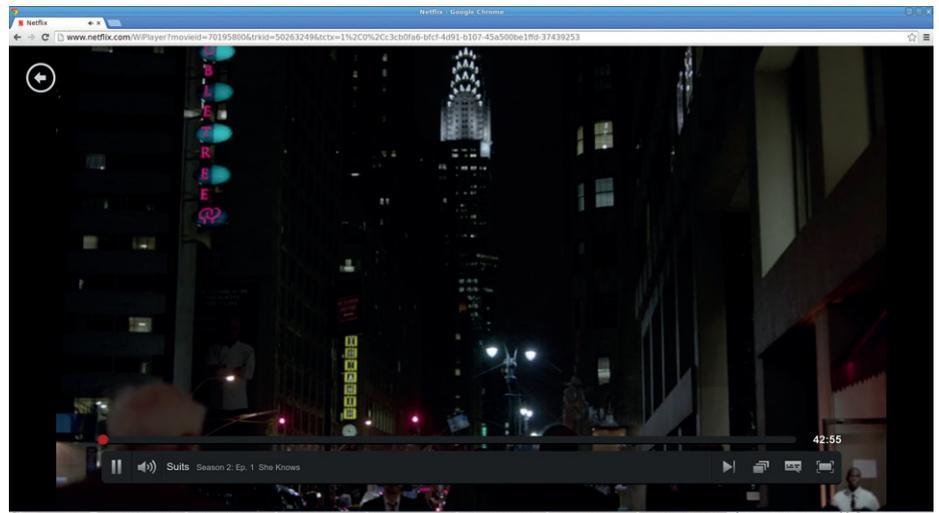
Desktop

You should feel comfortable in your computing environment.



We're sure that there's someone who likes how Windows 8 looks, but we haven't found them. Ultimately, the problem with the Windows desktop isn't that you may like or dislike how a particular version of Windows looks, but that you're stuck with the desktop environment that Microsoft thinks you should use. Some people like traditional desktop environments with a taskbar along the bottom and a menu in the lower-left corner, while other people like tiled window managers. Some people have embraced Gnome Shell and Unity, while others think they're an aberration of all that is good in this world. It doesn't really matter what you like – what matters is that it should be up to you to decide what desktop environment you like, not some multinational corporation that's primarily interested in profit.

It's not just looks that makes the Linux desktop better than Windows though. There's one thing that always shocks us about Redmond's OS: the complete lack of package management. If you want new software, you have to download it from a website and install it by hand. Of course, this means there's also no centralised update system: the software you download may or may not have a mechanism to stay up to date. Even if it does, it will be out of sync



Netflix now works natively on Linux using HTML 5 video. That's one less reason to dual boot.

with the other updates on your system, so you (as the user) are being constantly asked to download yet more software. And this is without mentioning all the intrusive browser toolbars and other ad- (and spy-) ware that can come bundled in these updates.

Some progress

It would be remiss of us not to mention that the situation in Windows is improving. With Windows 8, Microsoft introduced an app store. This is a centralised place that users

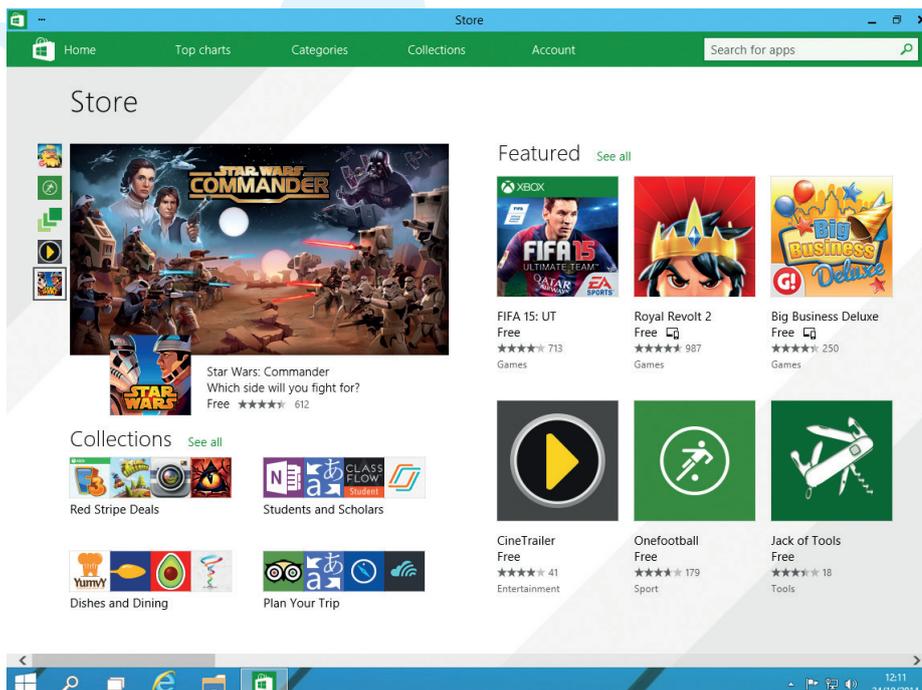
can go to to download quality software and be sure of its origin. Or, at least, that's what it should be. In reality, Microsoft's management of the store has meant that few software vendors have decided to use it. As a simple example, we searched for office software. There are many good pieces of office software available for Windows, from the open source suites (*Libre* and *OpenOffice*), to closed source suites (*Kingsoft* and *Softmaker Office*) to smaller products. The results of the search were (as ordered by the store)

- One Note (Microsoft's note taking application).
- Office Depot (A map showing the location of Office Depot stores).
- Office Evolution (A table showing which applications have been in which version of MS Office).
- Office 365 Garage Series (A list of videos for Office 365).
- Office Academy (A tool to help you use MS Office).

The list goes on in much the same fashion. There's almost nothing useful, and almost everything is a simple wrapper around online content. None of the useful office software we mentioned before is available on this app store.

Compare this to the list we got when we put the same search term into Ubuntu 14.04:

- Zoho Webservices Presentation.
- Zoho Webservices Spreadsheet.
- Zoho Webservices Wordprocessor.
- Extra Office applications (a menu for



The Windows store does have some good games, but otherwise it seems woefully barren.

Ubuntu Studio).

- Office Worker (a game).
- LibreOffice Base.

The list goes on to show the rest of the *LibreOffice* suite and other useful software. Admittedly, this list isn't the order we'd ideally put Linux office software in. What's more, the Ubuntu store is built on top of a package manager. The two pieces of software are often confused, but they ultimately serve different purposes. A software store (or software centre) is a place a company can publish software, usually accompanied by details about the software, screenshots and user reviews. A package manager, on the other hand, is a system for installing packages and keeping them up to date.

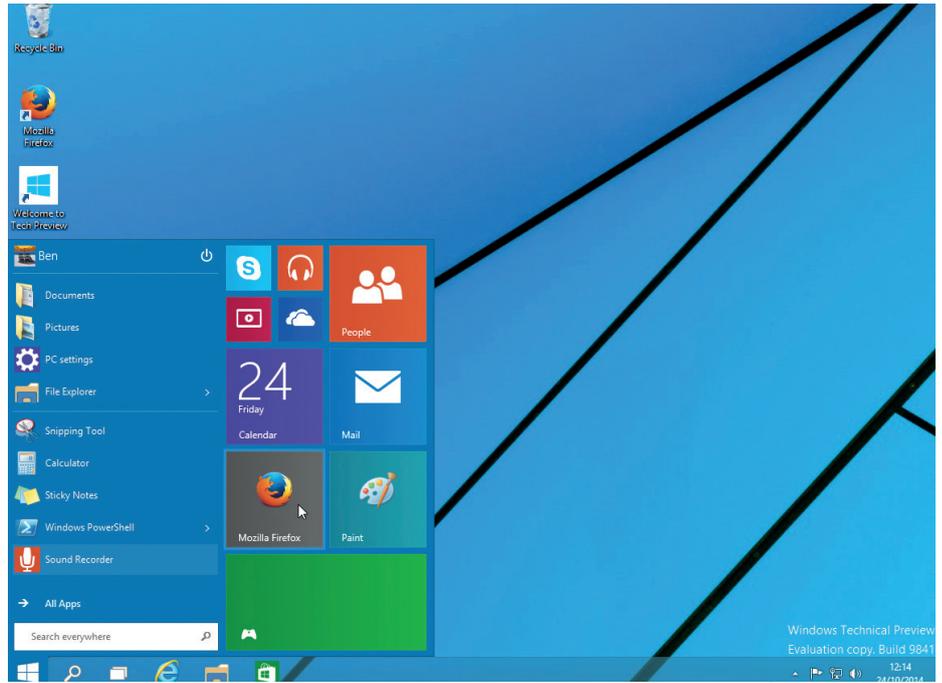
Microsoft has promised a package manager for Windows 10. However, it won't have the power of a Linux package manager. In fact, it will be little more than a command line wrapper around **install** files, and so won't really make the situation much better.

Gaming

Just a few years ago, if you'd dared to suggest that Linux was a good gaming platform, you'd have been laughed at. Now, industry leaders such as Gabe Newell are claiming this very thing. To understand why, you have to look a little at the history.

Traditionally, gaming has been done on two distinct types of machines: consoles and multi-purpose computers. Consoles are stripped down computers with everything not necessary for games removed. However over time, the hardware differences between games consoles and general purpose computers got smaller.

When Microsoft released the first Xbox console in 2001, it was really just a PC in a different box with a different interface. By releasing a games console, Microsoft became a games publisher. This put the



Windows 10 brings some sanity back to the Windows desktop after the clusterfail that was Windows 8, but users still only have one desktop environment to choose from.

company that created the OS in direct competition with many companies that created software for the OS. However, this wasn't the only factor pushing games towards Linux.

As games consoles were changing, so too were regular computers. PCs became the dominant platform, and PCs are more hackable than the computer that came before them. You could replace the CPU, motherboard, storage, or just about anything else in a PC with parts from a wide range of manufacturers. You could even build your own from parts.

DIY ethos

This hackability was something that PC gamers took to heart. Some people relentlessly pursued ever-higher frame rates

using more and more advanced hardware. Others built weird and wonderful machines out of old PCs with just enough muscle to run modern games. Even games started to become more customisable, with some publishers actively encouraging players to create new levels and artwork. However, on top of this was always Windows, and this has become more and more locked down with each version. Ultimately, with a community that was used to ever-increasing hackability and an OS that was becoming ever more restrictive, something had to give.

The final tipping point came in 2012 when Valve released its Steam games distribution platform for Linux. Since then, Valve has been encouraging games makers to release their titles for Linux.

Linux is a natural choice for PC gaming. It's the only OS that fully supports the hackable philosophy of PC gaming. The number of games available has exploded in the last couple of years with three new major sources of games (Steam, GOG and Humble Store), as well as the faithful stores that have been selling Linux games for years (such as Desura).

Hardware support

Unfortunately, there are some manufacturers that only support Windows. There are even some that actively hamper development of Linux drivers by people in the community who want to create open source drivers. The situation is getting better, but it's far from perfect.

However, this doesn't mean that Windows has better drivers than Linux. The situation is a little complex, and there are a few things you need to consider. Most Linux drivers are part of the main kernel project. This means that they're automatically available without the user having to download anything additional. It also means that they're subject to quality checks, and are updated

by kernel developers when changes are needed to ensure they work with the latest kernel.

Hardware drivers on Windows, however, are developed entirely by the hardware manufacturer, and usually included on a disk that comes with that hardware. On the one hand, this is convenient, but on the other hand, it means that the hardware manufacturer can force you to install all sorts of weird software just to use the hardware. The manufacturer may also not update drivers for older hardware for newer versions of Windows, meaning that you can get stuck with either a computer you can't upgrade or hardware you can't use – both problems that just don't arise with Linux.

TOP 3 REASONS LINUX IS BETTER ON THE DESKTOP

- 1 Flexibility
- 2 Games
- 3 Hardware



Power users

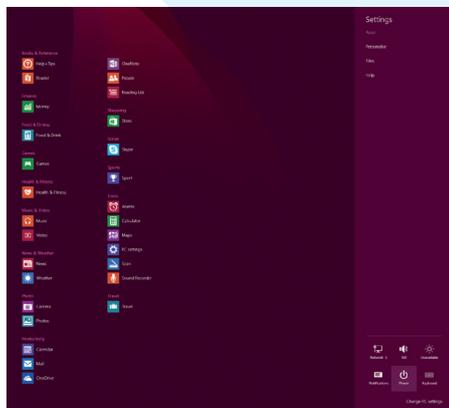
There's only one operating system for those who like a little power.

Microsoft is as ubiquitous as x86 processors. It could even be argued that it owes its success and dominance to their existence, going all the way back to 1981 when Microsoft's PC-DOS and MS-DOS were sold for use on IBM PCs and non-IBM (yet still compatible) PC clones. And while those very early machines used Intel 8088 CPUs, they were closely related and quickly supplanted by the 8086 as the market exploded, creating the foundations for what is Microsoft and the PC market today.

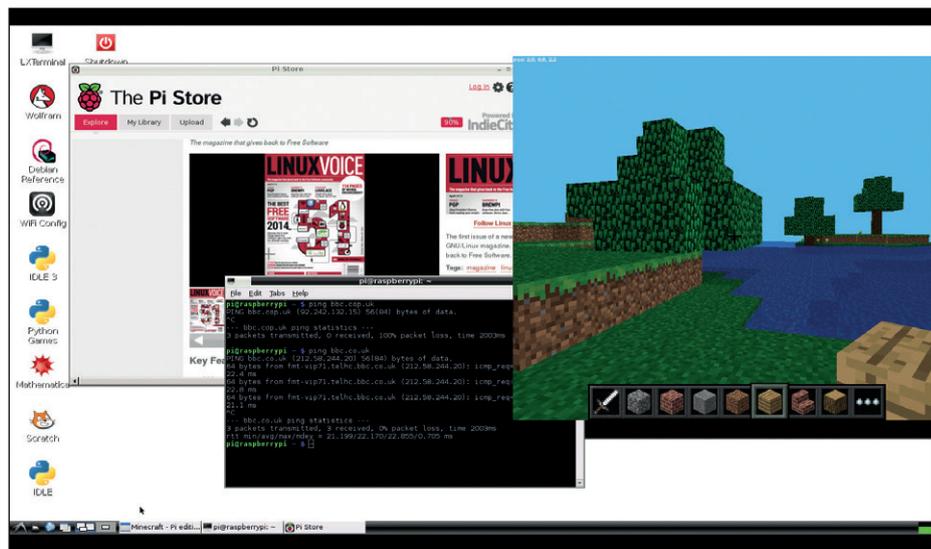
When you look at it this way, it's not surprising that Microsoft hasn't embraced other architectures, and the much anticipated ARM version of Windows 8 has done little to change this. Windows RT, as it is known, looks and feels like Windows 8 but its transformation to ARM has more to do with providing something for the tablet market, rather than new territory for Windows developers; Windows RT is fundamentally the Windows 8 experience running on slower hardware, which misses the point behind Linux's incredible success on other platforms.

“From toasters to robots, games consoles to weaponry and space stations, Linux is everywhere.”

Linux has never had any kind of problem running on non-x86 hardware, and we're sure that's because it's open source. When anyone can use this source code, almost



If you think Gnome has shutdown problems, look at Windows 8. Move the mouse somewhere then click on 'Settings' – only then will you find the fabled shutdown button.



Linux works on all kinds of hardware, which means the skills you learn on the desktop are incredibly transferrable – here's Raspbian running on an ARM-based Raspberry Pi, for example.

anyone will try and get it running on almost anything – from toasters to robots, games consoles to weaponry, space stations to sous-vides, Linux is everywhere. Debian has official ports for ARM, MIPS, PowerPC, IBM's S/390 servers and Sun SPARC, while

substantial projects that have taken a piece of consumer hardware and subverted its Linux kernel into a much more open and ambitious project. If a piece of hardware is using Linux, you've got a far better chance of being able to get access to its innards – whether that's unlocking frequencies in a commercial oscilloscope (a Rigol DS1052E hack), adding web access to a digital video recorder, or just putting XBMC on your favourite ARM-based mini-PC.

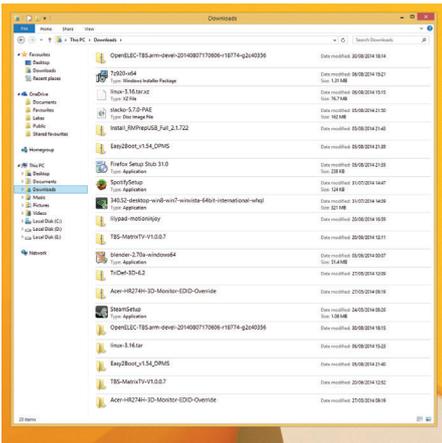
the unofficial ports still include code for processors such as the venerable Motorola 68000, for instance. But it's the ARM ports that have perhaps had the biggest impact, because the ARM 'System On A Chip' packaging has transformed the embedded market – a complete Linux PC that fits into a tiny low-powered space, whether that's Android on smartphones and tablets or Raspbian running on a Raspberry Pi.

Regardless of the architecture, if you're using GNU/Linux, you're using the same underlying operating system. That means you can use many of the same skills you learnt for playing media on your desktop to create a media streaming device that attaches to your television. Or the same skills you used to install OwnCloud on a virtual private server. This also expands to development and the ease with which you can hack your own software and hardware. Linux is unparalleled for this. There are many

Development

Writing code isn't for everyone. But software development is not only vital for native development, it's also vital if a platform is to succeed as a working environment for cross-platform development. Visual Studio and Microsoft's associated toolkits actually provide a world-class development environment just as vital to many businesses and corporations as *Microsoft Word*. The use of .Net and DirectX have created industries in their own right.

But over the last decade, open source toolkits and development models have transformed the industry. It's not just about open source compilers, or the GNU toolkit, or that you can even code on a Raspberry Pi. It's about creating a shared resource – something that you couldn't have done with Microsoft's development environment 15 years ago. Open APIs are the backbone of the internet, whether that's Google probing

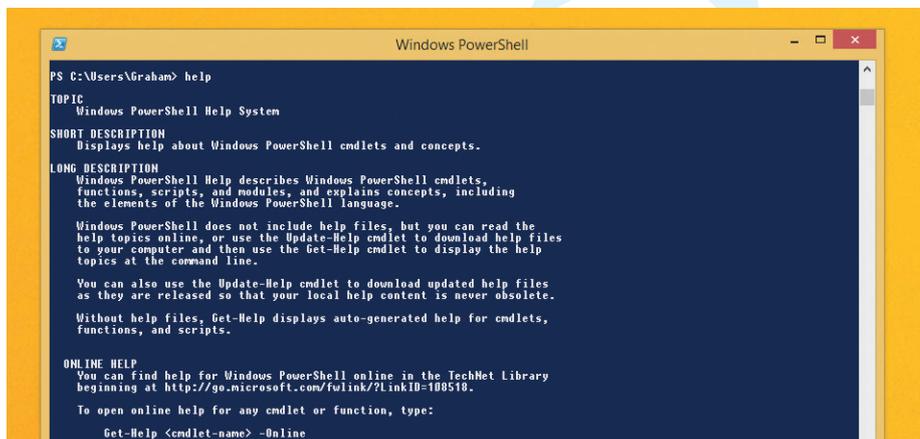


The *Windows Explorer* file manager can also be replaced, but because most users don't bother, your options are limited.

your emails for advertising potential, or for the latest responsive web frameworks.

Linux is at the heart of this. It's the native environment for web site creation, and with IDEs like Eclipse, the native environment for many kinds of cross-platform development too. And what's more important is that with many computing curriculums changing to focus more on programming and hacking your own hardware, the combination of Linux and something like the Raspberry Pi is unbeatable. It's the best kind of lock-in, because we're getting the next generation of coders exposed to open source and Linux, and once you've experienced that freedom, it's very difficult to move to a development environment that offers less.

The command line is still simply the easiest way to perform many kinds of tasks. But in its attempt to distance itself from the old DOS days, and to emphasise the point-and-clickiness of Windows, the DOS prompt became a 90s relic. After being in development for a couple of years, this



PowerShell is a great addition to Windows. But it's not a replacement for the Unix mentality of *Bash*.

OpenWrt

OpenWrt is one of the best known custom firmware replacements for commercial hardware (and even the Raspberry Pi), as it replaces the default operating systems for many standard routers. This is a vital job, not only because it adds a layer of trust that only open source audited code can provide, it also adds features often restricted to far more costly devices. There's an SSH server, for example, which is very handy when it's built into your router, as it enables you to create simple VPN tunnels to your network, and there are many other packages that can be installed, much like you can with a regular Linux distribution. But even in a commercial setting, OpenWrt is used to replace enterprise-grade hardware costing £1,000s, even if it's just to use the traffic shaping features of the new firmware. And all of this is thanks to Linux being open source and because Linksys had to release its own firmware when it was built upon the original GPL code.



OpenWrt started life on Linksys' original WRT54G (the later WRT54GL show here), and used the GPL code released by Linksys [image: CC BY-SA Vidarlo ENWP]

This couldn't have happened any other way, and it's not even the only time this happened with wireless routers. Another custom firmware, dd-wrt, is also available for certain Linksys models and offers a very similar feature set, and there are even more choices for other hardware platforms/routers.

finally changed with Windows 7 when the 'Power Shell' became an integrated part of the update. *PowerShell* is a much closer approximation of the Linux command line, and has succeeded in becoming an essential too for Windows sysadmins. There's even an update for Windows 10, in the form of keyboard shortcuts. But it's still a vastly different beast to the Linux/Unix shell.

It's about choice...

Most importantly, with Linux you have a choice. *Bash* is installed by the majority of distributions, but every user is free to change their default command environment, or change whenever they want, as each shell does things differently. *PowerShell* could or should be another one of those options, because it's actually good at some very Windows/Microsoft specific tasks. It's excellent at piping output and keeping the

context of the data you're working with. It works brilliantly with Microsoft's object models and is a highly useful extension to the latest version of Windows.

But it's not a Unix shell. For example, tools such as *Emacs* or *Vim* were designed to maximise the Unix-alike environment they were running in, and it's the same for the vast majority of tools you can run from any command line. You can still perform any task on your computer from the command line, regardless of the GUI environment you might have installed. *PowerShell* isn't trying to compete with that functionality, which means Windows is still missing out.

Which brings us to another point. With Linux, you're never forced to upgrade, or have an application force an upgrade, or be left with files that don't load into anything you can't afford to buy. And many proprietary formats change from one version of an application to another, meaning you'll need the same version of *Photoshop*, or the same version of *Word*, to get the same data when you share a file. This just doesn't happen with open source because the emphasis is always on open standards and interoperability – it's why, for example, *LibreOffice* supports many more formats that *Microsoft Office*.

TOP 3 REASONS FOR POWER USERS

- 1 Hackability
- 2 The degree of control
- 3 Choice



Interoperability

Can't we all just get along...

Microsoft Windows is and always has been widely used in business. And this isn't likely to change. Many companies, especially older ones, have a long history with the software, its capabilities and its licensing. And companies don't like change any more than the users do, especially when Microsoft does create rather good business software. Ignoring the usability and graphical overhauls, Windows is a perfectly good operating system when you need to get things done, and ignoring the cost, there's a good reason its business-focused applications are the backbone of its business. They're industry standards because they work well.

But there is a change coming, and it's one we think Microsoft is going to find increasingly difficult to attack. As the desktop becomes less important, and users now have more than one device, more than one machine, and more of one way of working, issues besides functionality and performance become more relevant. In particular, open computing and data standards are finally being recognised as being important, after decades where proprietary formats have ruled in local government, health and education. In the UK, for example, government documentation in its static form has to be provided as PDF/A, which is an open format that can be read by lots of different

“After decades, open computing and data standards are finally being recognised as important.”

kinds of software. In Switzerland, the open source community has just crowdfunded approximately €8,000 to pay for *LibreOffice* developers to add digital signatures to PDF/A documents. This is a requirement for PDF documents to be legally binding, and is another great step towards making digital documentation and facilities available to as many people as possible (the campaign is still accepting money until the end of the year: <http://wilhelmtux.ch/?MID=11&PID=93>).

Of course, none of this has anything to do with Linux specifically, but by using Linux you are ensuring this interoperability



The Free Software Foundation is running a campaign to highlight the free alternatives to Adobe's default PDF reader and icons – that's why we're supporting their use on LinuxVoice.com.

and openness remains relevant. By getting other people to use Linux, you're keeping this forward open momentum going. It's the same reason Microsoft fought so hard, and succeeded, to ratify its own Office Open XML format, despite an open format (ODF) already existing. It needed its own formats in place in order to control the way people shared documents created and edited with

its own software, regardless of whether that meant the format was open or not. And of course, it was always going to be in Microsoft's best interests to do so. There's nothing wrong with that – it's the best possible way of safeguarding its business interests. But when open standards make all of our lives easier, Microsoft's stance isn't always going to be in our best interests.

To fork or not to fork

Another more tangible advantage that Linux has is the power to fork the operating system. Forking refers to the act of taking the source code, copying it under the terms

of an open source licence, and continuing to develop – usually with a different emphasis, or maintainer, or community. The ability to do this is enshrined within open source, and it's completely different to the closed and proprietary development model of companies like Microsoft and Apple. To the greatest extent, they view their code and their software as their intellectual property, and increasingly, they're not selling us an application, they're selling us the licence to use their property for a period of time.

Forking has resulted in lots of duplicity and choice, which isn't always a good thing. But it means that if users don't like the way Ubuntu is going, they can create a fixed version of Ubuntu. It will either thrive or fail depending on whether other users agree, and that's why we have distributions like Mint. It also safeguards the software – *LibreOffice* is a fork of *OpenOffice.org*, created when the future of the original project was unclear. X.org, the display system used by most installations, is a fork of XFree86, from a time when development had stalled and other operating systems were advancing quicker than Linux.

It's a completely different kind of development, and one we think results in simply better software. It's also an attitude

that rails against the burgeoning patent war of the big software creators, along with the constant battle to keep software patents out of Europe.

Which neatly leads us on to education. We're going to argue that skills learnt on Linux are more transferable, and that's why it's important that educators use Linux or open source software. If you learn how to create digitally signed PDFs with *LibreOffice*, for instance, you can now use *LibreOffice* on many different types of platform and from many different types of computer.

Education, education, education.

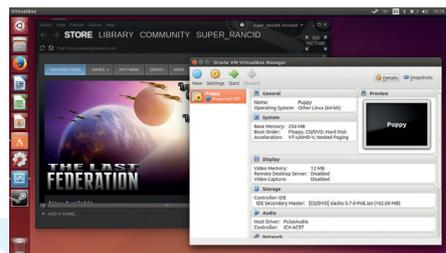
But this also works at a lower level. Many educational authorities are changing their computing curriculums to reflect the way we now use computing and to hopefully emphasise the need for computing science. This September in the UK, for example, marked the start of a new curriculum that throws out much of the style curriculum. Or in the words of the UK's Education Secretary at the beginning of the year, "ICT used to focus purely on computer literacy – teaching pupils, over and over again, how to word-process, how to work a spreadsheet, how to use programs already creaking into obsolescence; about as much use as teaching children to send a telex or travel in a zeppelin."

He then explained how he wanted it to change, "Our new curriculum teaches children computer science, information technology and digital literacy: teaching

Convergence

Both Microsoft and Apple are trying hard to make their users an integral part of their services, and part of this strategy is convergence – multiple devices acting as one, like being able to use *Word* on both your tablet and your desktop. Or your tablet becoming your desktop. There are Linux projects and distributions promising to do the same, with Ubuntu's imminent phone being perhaps the best example. But another aspect to convergence is the shared data services for your device, whether that's for the documents that you edit, the photos you take or the applications you purchase.

As with everything else, Linux differs because you have a choice. You may choose to run *OwnCloud* on a server somewhere and manage your own shared files and devices – there are *OwnCloud* apps that run on both Android and iOS devices, so this works even with mixed hardware. Or you can even choose to use the



Ubuntu has re-used many of the same ideas in its desktop, netbook and smartphone versions.

same services provided by Microsoft and Apple. The great thing is that your operating system won't be held hostage to this choice, and neither should your data. And you also get to choose where and when you use these services, whether that's on your own hacked tablet, or BusyBox running on a NAS device, or a router with a custom firmware. And we think that's the best definition of convergence.

them how to code, and how to create their own programs; not just how to work a computer, but how a computer works and how to make it work for you."

Tomorrow belongs to us

There's only one operating system we can think of that teaches "how a computer works and how to make it work for you," and that's Linux. Better though, is that these skills won't lock you into Linux. They'll give you a lifetime of perspective whenever you use technology, whether that's a cash dispenser or the latest Macbook Pro. This is

likely to be why the Raspberry Pi has been such a success, and why it uses Linux as the default operating system. It doesn't matter that it's Linux, just that it enables you to do whatever you need to achieve a specific educational goal.

TOP 3 REASONS LINUX WINS AT INTEROPERABILITY

- 1 Standards compliance
- 2 Education
- 3 Convergence



TWELVE REASONS WHY LINUX WINS

You knew it was a forgone conclusion.

After all that, it feels like we've only scratched the surface. What's also interesting is how things have changed. Ten years ago we may have started a comparison like this as a feature-by-feature comparison of the two operating systems, as if the desktop was the only way one operating system could ever come to dominate over another – or win, as we'd have put it at the time. Many of us still use Windows, and it still does a great job in the millions upon millions of places it's installed and used. It's a huge accomplishment and a testament to the skills of the corporation behind it.

But side-by-side comparisons are no longer relevant. We hate using the word, but

our comparison has become more 'heuristic' for a good reason; technology is now shaping almost every aspect of our lives and it's no longer about one company or operating system. It's about your data, control, freedom and choice, and these are ideas that don't align with our old way of thinking.

They're also ideas that we feel strongly about, which naturally leaves us with only one choice of operating system. We know we're preaching to the choir when we say Linux wins in so many different ways, but it's important to remember why, and just why GNU, Linux and Free Software are getting stronger and more relevant with each passing decade.

TOP 3 REASONS LINUX IS BETTER FOR . . .



USE ONLINE

- 1 Servers
- 2 Security
- 3 Flexibility



USE WITH LINUX

- 1 Flexibility
- 2 Games
- 3 Hardware



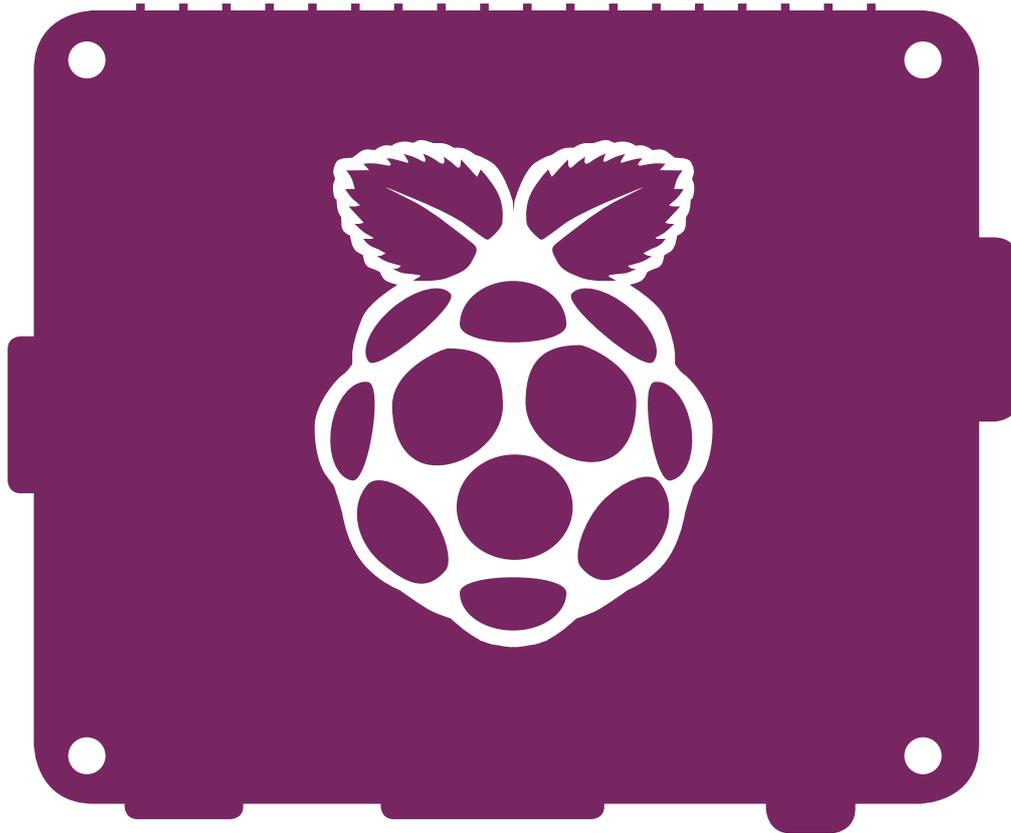
POWER USERS

- 1 Hackability
- 2 The degree of control
- 3 Choice



INTEROPERABILITY

- 1 Standards compliance
- 2 Education
- 3 Convergence



TOP MARKS FOR RASPBERRY PI

Russell Barnes speaks to the founder of the Raspberry Pi Foundation to find out why he thinks the Model A+ is their most exciting release yet...

A+

Eben Upton came up with the idea of an affordable credit card-sized Linux PC while working as the Director of Studies in Computer Science at St John's College, Cambridge. Part of his role was to manage and monitor undergraduate admissions, something Cambridge appeared to be fast running out of.

"It wasn't a case of me waking up one day wanting to make a credit card-sized computer, but waking up one day and realising we've got no computer science students and wondering why," explains Upton. "Ultimately we came to the conclusion that there was a distinct lack of programmable

computers for kids and we started putting some thought into what we could do about it."

As he talks Eben brandishes the latest offering in the Raspberry Pi line-up, the Model A+. It looks rather

different to its predecessors, almost as if someone's taken a hacksaw to a Raspberry Pi and lopped an end clean off. As it turns out, that's almost exactly

"The model A+ looks as if someone's taken a hacksaw to a Pi and lopped an end off."

what happened, but there's a bit more to the story than that...

For a start, this isn't the first time the Raspberry Pi's winning formula has been tweaked. Earlier this year the Foundation launched the B+, an improved version

Eben Upton: Quickfire Q&A

Q What's your favourite thing about the model A+?

A It's amazing what ergonomics can do for the popularity of a product. One of the things people love about the B+ is the fact it looks nice. It's got the rounded corners, the mounting holes and the connectors are all lined up with the edge of the board – it's just nicer.

I'm really obsessed with the rounded corners [laughs]. It's a little thing and it's really easy, but it makes such a difference on the A+.

Q Does the \$20 make the Pi more of an Arduino competitor?

A We're just about in the range of official Arduinos in terms of price. Of course we're not an Arduino competitor. I don't think there's a great deal of overlap between what the Raspberry Pi is the right product for and what the Arduino is the right product for. The Arduino still consumes less than power than an A+ and it has analog inputs. The Raspberry Pi on the other hand drives a display and has more processing power. There's still a

massive gulf between the two products.

Q Will the Raspberry Pi Compute Module ever take off with industrial partners?

A We don't want to push people to the Compute Module until they're ready. I think the Compute Module is well suited to industrial solutions, so if someone needs 10k of something it's definitely worth doing a small amount of R&D so you can go for a Compute Module, but it takes time to make that transition.

of its 3.5 million-selling Model B. It's the board Upton refers to as the 'deluxe model', because while the Model A is the Foundation's affordable flagship Pi priced at \$25, the B comes with the added convenience of extra USB ports and Ethernet networking for an extra \$10. The dream was to produce a \$25 computer powered by open source software, but it seems the vast majority of geeks and educators were more than happy to pay an extra ten bucks for the privilege.

Thanks to the massive success of the Model B, the team employed by the charity has positively ballooned over the last 12 months (even so the entire workforce can be counted on two hands and one foot). One of those new employees was Director of Hardware, James Adams, who was tasked to taking over from founding member Pete Lomas to create the Raspberry Pi Plus line.

With a combination of user feedback and raw common sense, Adams utilised the new economies of scale and reputation the Foundation had achieved to really go to town with the Model B+ improving the board layout, adding more General Purpose Inputs & Output (GPIO) pins, doubling the number of USB ports and massively improving power consumption among other things. And all this happened at the same retail price as the original Model B.

It was just the ticket too. The B+ became the fastest selling credit card-sized PC in the world almost

overnight. "Even though it was only a couple of weeks from when the B+ came out, we sold more Raspberry Pi's in that half of July than we've ever sold in any single month before," enthused Upton. Today the Raspberry Pi is selling over 100 thousand per month and the Foundation is on the brink of selling its four millionth Linux-powered PC.

Do we need a Model A?

With the Model A selling less in its entire lifetime than the Model B sells in a month, it wouldn't be outside the realm of reason to wonder if the Model A line is worth

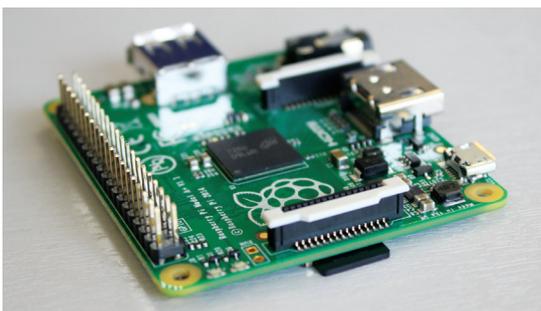
Original Model B goes back into production

The old Model B is once again rolling off the production line at Sony's facilities in Wales.

"Obviously when we started the Model B+ we wanted to make sure our channel partners weren't stuck with lots of Model B," says Upton. "We've got a lot of small businesses that started around the Pi and if you leave them with lots of old inventory it could cause them lots of problems. We told them all about the B+ and made sure they could move their old stock before we put the B+ on sale.

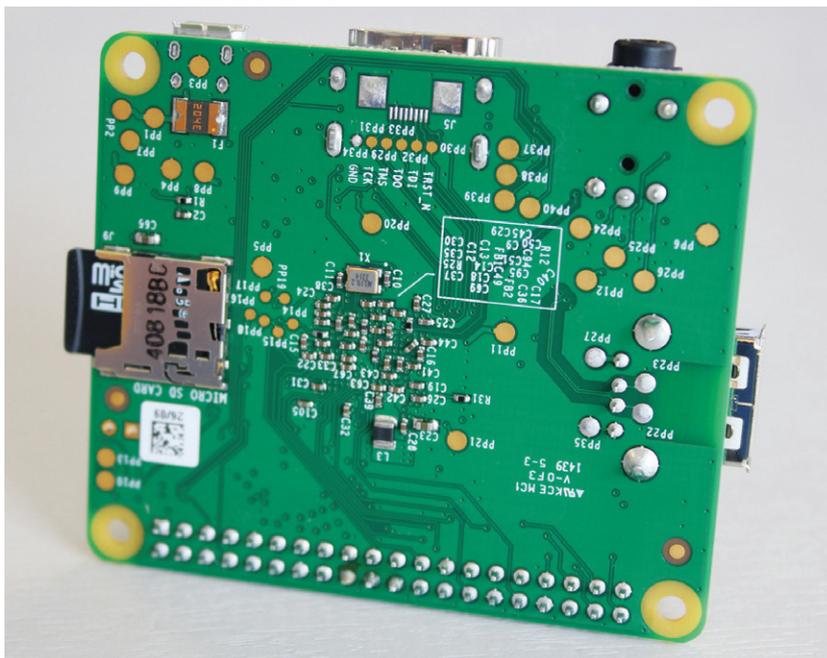
"After putting the B+ out there what we found was that a lot of people who were putting the model B into their products were saying, 'actually we're pretty happy with the Model B, can we keep buying it?'. It's great that we've got industrial customers that turn around and ask for 10,000 more Model Bs."

But what about the original Model As? "We're definitely not going to make any more As. Industrial demand for As hasn't been anywhere near as strong, so I really don't want to restart the A line."



The key benefit of the Model A+ isn't just the low price – it's the low, low power consumption.





Check out those corners – see how round they are.

continuing at all. Indeed, the \$25 credit card-sized PC for schools is barely in any schools, despite there being a good number of more expensive Model Bs to be found in the education sector.

As it turns out, it's only really hardcore hackers that make any real use of the Model A. People like Dave Akerman (www.daveakerman.com). Dave regularly attaches Model A Pis to weather balloons so he can take pictures with the Camera Board from an altitude of around 35km – right on the edge of space. With projects of this magnitude every milligram and milliwatt really count and the Model A's lighter load in terms of weight and power consumption makes it the ideal candidate.

Eben is the first to admit the Foundation failed to communicate the benefits of the Model A beyond its cheaper price, and it's something the team are determined to rectify with the new Model A+.

"It's easy for people to look at the Model A and think it's just a cheaper variant of the B. When they look at it like that they might as well just go for the deluxe

model since it's only an extra \$10," explains Upton. "I feel like some people missed out on why the lower-power model like the Model A can make sense. If you're building something with robotics, or essentially any project that doesn't need Ethernet networking, it's a great fit."

Eben also thinks it would make a mockery of the original \$25 computer promise if they didn't continue with the Model A: "It's also really important to us because it's our flagship product. It was our original stake in the ground and where it all started."

Chopping the end off the B+

While some tech firms would be nervous to have another try at a less successful model, the Foundation is very excited about the release of the A+. A look over the specs shows it's a big improvement over the flagship Raspberry Pi, but the groundwork for the A+ started when the B+ was still the drawing board.

Upton says: "James Adams came over to see me [with a Model B+] and said 'you know we can chop the end of this board off, right?!'"

It transpired that Adams actually designed the Model B+ with the Model A+ in mind, making it a trivial board design tweak to shave quite a large amount off one end of the board for the A+.

"Where the original Pi had *ad hoc* mounting hole positions, the B+ has these nicely positioned square mounting holes on the body of the board, then an extension on the right that contains the Ethernet and USB ports, which are 'outboard' of these mounting holes," continues Upton. "James basically explained that we can chop the board off at the mounting holes and find room on the board for a single USB connector, meaning we could make an A+ board that was markedly smaller than the B+."

It certainly feels impressive in your hands. The mounting holes are in the same place as the B+, but with the shorter board they're located in all four corners, making it a shade off being totally square.

Since it has the same improved GPIO as the B+, it's also still compatible with the new Hardware Attached on Top (HAT) standard for Raspberry Pi add-on boards. In fact, the outline of a Model A+ and a HAT board is identical, making an A+ with HAT a very compact and mobile combination.

"We set about making a mock-up of the design and we all agreed that it made a really quite attractive product," continues Upton. "It really adds something to the mix and it adds something to its uniqueness too. It's cheaper than the B+, it consumes less power, but now it's quite a bit smaller too."

Not just smaller in the X dimension either – it's almost half the height too. Until it was replaced in the Model B+ by a dual-purpose 3.5mm audio and composite jack, the Raspberry Pi's board height was dictated by the massive (not to mention bright yellow) composite video port. Now the limiting factor on the height of the board is the low-profile USB port, bringing the Z dimension down from 21mm to 12mm.

Model A+ essentials

With only one USB port, getting by with a Model A with anything more than an SSH connection over Wi-Fi can be tricky. Here are a few add-ons designed to help...

■ **Broadcom WiFi adapter & 2 port USB hub** – £9.99 (www.pi-supply.com)

The fact that the drivers for Broadcom's range of Wi-Fi adapters have only just appeared in the Raspberry Pi's official distro, Raspbian, is a minor embarrassment, but perhaps by way of making up for the oversight you can now buy this two-port USB hub with built-in Wi-Fi. OK, it's not going to win any awards for design, but you can use it to turn your Model A+ into B+ at a whim.

■ **Riii Miniature Wireless USB Keyboard with**

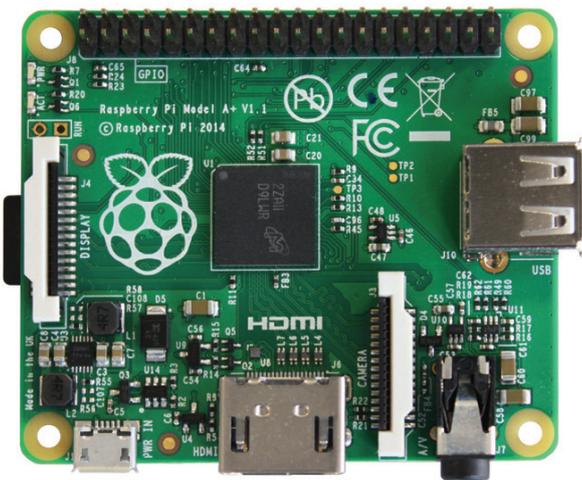
Touchpad – £26 (www.thepihut.com)

This tiny wireless keyboard with touchpad will give you decent control over your Model A+ with a full qwerty keyboard, a built-in touchpad, backlit keys, multimedia controls and even an integrated laser pointer.

■ **USB sound adaptor for Raspberry Pi** – £3.99 (www.modmypi.com)

Want to use your Model A+ as a decent audio player or a voice recorder? Although the A+ and B+ have improved audio support, there's no microphone input and the audio it outputs is still low-definition. The only way around either of these problems is to use a USB audio card, like this affordable little fellow from ModMyPi.com.

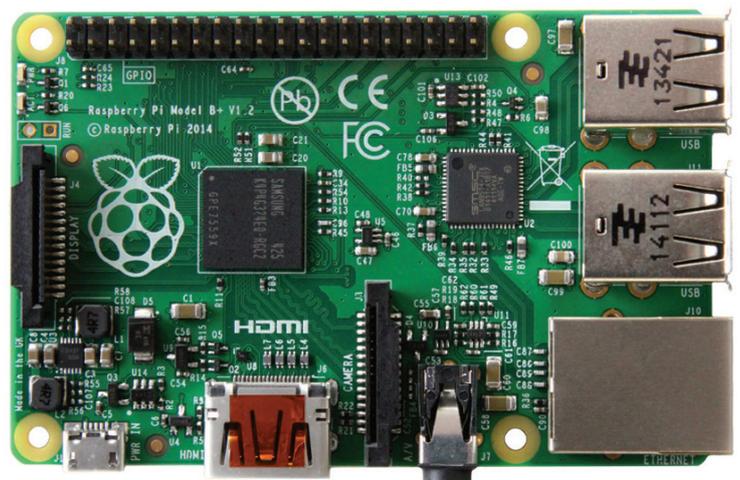
Side by side How the Model A+ and B+ stack up

**Model A+ specs**

Price: ~£15/\$20
 Dimensions: 65 x 56 x 12mm
 SOC: Broadcom BCM2835 (700MHz)
 Memory: 256MB RAM
 Networking: None

Expansion:

1 x Micro SD card slot
 40 GPIO pins
 1 x USB port
 Power consumption:
 200mA under load

**Model B+ specs**

Price: ~£27/\$35
 Dimensions: 85.6 x 56 x 21mm
 SOC: Broadcom BCM2835 (700MHz)
 Memory: 512MB RAM
 Networking: Ethernet 10/100

Expansion:

1 x Micro SD card slot
 40 GPIO Pins
 4 x USB ports
 Power consumption:
 370mA under load

(left, Model A+; right: Model B+)

Like the Model A, the A+ features just one USB port, 256MB RAM and no Ethernet port. Besides this (and the size of the board) Eben says it's 'electrically identical' to the Model B+. This means the A+ now benefits from the B+'s improved power chain, meaning you don't need the mother of all micro USB power adaptors to keep your Pi running or a powered USB hub to enable the use of USB storage. On top of that, if you're looking to run a project from batteries you're in for a massive boost in longevity.

We've yet to put the Model A+ through the wringer on a benchmarking test bed, but we know it draws about 200mA from the 5V power with a keyboard plugged in, running 'hello_teapot' over a HDMI monitor. Compared to the already impressive 370mA from the Model B+ it's head-turning stuff.

The \$20 credit card-sized PC

The Raspberry Pi Foundation has always been ambitious when it comes to price, and the A+ already just about the cheapest single-board computer it's



HDMI out, and the potential it brings for displaying video to your television, is a strong factor in favour of the Pi.

possible to make. That said, the Foundation hasn't rested on its laurels with the A+. In fact, they ripped up their own '\$25 computer' rule book.

"I had a great conversation with [Google boss] Eric Schmidt last year when Google gave us a large donation to help us distribute Raspberry Pis to school children," says Upton. "We had a really good conversation and we were talking about price and performance among other things and he said to me 'try and be as cheap as possible... try and get as close to free as you can.'

"I found it really inspirational to have someone like Eric not just say 'well done; you've got a great product', but 'why aren't you asking yourself how to be cheaper?'"

It wasn't long until the Foundation ran off the prototypes of the board and he and the team really took to the new design. "It's just a great product," exclaims Upton. "I almost don't care how many of these we sell. The A+ went from something we knew we had to do to something that we're actually really enthusiastic about.

"It gives people a really low-cost way to come and play with Linux and it gives people a low-cost way to get a Raspberry Pi. We still think most people are still going to buy B+s, but it gives people a way to come and join in for the cost of 4 Starbucks coffees."

According to Upton they're really bumping up against the limits of how much you can build a significantly high-tech product for and not have people lose money in the process.

"It's already about the cheapest thing you can get that does this kind of thing by some margin, but we'll never be complacent about that," he concludes. ■



LITIGATION VS FREE SOFTWARE

A flurry of patent lawsuits has affected the landscape for open source software developers and businesses operating in the United States. Adam Saunders examines the state of play.

The United States is a popular region for patent litigation for a few reasons. There are some courts, such as the Eastern District of Texas, that have earned a reputation for being “plaintiff-friendly” when it comes to patent cases. That is, if someone brings a patent infringement lawsuit there, they’re more likely than not to win it. The payouts are also pretty high in the United States for a victorious plaintiff; awards can be in the hundreds of millions of dollars, with the highest award given weighing in at over \$1.6 billion (US).

As an individual admitted to practice law in my jurisdiction (Ontario, Canada) and an open source enthusiast, I have the background in the English common law system (a basic foundation shared by most English-speaking countries) to analyse and understand these cases.

Rockstar v Google et al

A ruling in this case could chill commercial efforts around the open source Android operating system.

Rockstar Consortium Inc., a non-practising patent holding company (sometimes known as patent trolls) owns a large number of patents. Rockstar was formed by a group of major IT corporations including Apple, Blackberry, Sony, Microsoft, and Ericsson to buy patents collectively as part of an auction of the Nortel Networks bankruptcy sale in 2011. Nortel Networks was a Canadian telecommunications corporation with operations around the world, making routers, computer hardware, and software. Over its lifespan, it filed for thousands of patents relating to its work.

In late 2013, Rockstar sued essentially all the major Android phone and tablet OEMs, including Samsung, Huawei, ASUS and Google for patent infringement

relating to Android. The patents covered a variety of areas, including one that could read on natural address translation, one that could read on notification shades in mobile devices, and one for GUIs to handle VPNs. For example, the drop-down shade that the user pulls down on Android phones and tablets to check your notifications implements ideas that Rockstar claims to own.

Google fought back in December 2013 to try to get the case pulled out of the plaintiff-friendly Eastern District of Texas and switched to the Northern District of California, by requesting a declaratory judgment of non-infringement. That is, asking the California court to declare that Google isn’t infringing. After almost a year, and a formal request made to the US Court of Appeals for the Federal Circuit, Google did manage to get the case switched to California from Texas; a major victory.

Huawei settled confidentially with Rockstar in January 2014. Samsung, HTC, ASUS, LG, ZTE, and Pantech would fight on; they moved to get the suit

dismissed on a jurisdictional technicality.

The Northern District court ruled in April in favour of Google’s motion to move the jurisdiction

there. But the Eastern District of Texas court refused to accept this, so Google asked the Court of Appeals for the Federal Circuit to move the case to the Northern District.

With Rockstar’s business model based solely on extracting money from patent licensing deals or lawsuits, this lawsuit may go on for years to come. We’re still waiting for the technical merits of the case to be assessed! Open source developers should follow this case, as the results may encourage or discourage commercial development of Android-based devices.

“Rockstar’s business model is based solely on extracting money from patent licensing deals.”

Octane v Icon and Highmark v Allcare

The rulings in these cases dealt significant damage to patent trolling in the United States.

The issue in Octane – a battle between two fitness equipment companies over patents on elliptical machines – was the standard by which a “court in exceptional cases may award reasonable attorney fees to the prevailing party.” The Supreme Court of the United States held unanimously in favour of the plaintiff, Octane, to the benefit of those fighting against patent trolls.

Supreme Court Justice Sonia Sotomayor, who wrote the decision for the court, began by tracing the history of the rules for attorney’s fee awards in patent litigation. The most recent change to the rules, Section 285 of the Patent Act, essentially inserted two words – “exceptional cases” – into those rules. Sotomayor noted that the Supreme Court of the United States had previously ruled that those two words merely clarify the rules:

Following the addition of an appellate court for all patent matters in the US – the Court of Appeals for the Federal Circuit (CAFC) – in 1982, the status quo was largely upheld for over 20 years; that is, “the Federal Circuit [...] instructed district courts to consider the totality of the circumstances when making fee determinations under §285” (page 5). But when the CAFC came across a particular case nine years ago – Brooks Furniture vs Dutailier – it decided on its own to implement a new standard: a defendant could only get attorney’s fees if the lawsuit was done “in subjective bad faith and [...] [it was] objectively baseless (page 8).

The Supreme Court has now cast aside that restrictive standard. After looking at dictionary definitions of “exceptional”, SCOTUS decided on this standard:

That an ‘exceptional’ case is simply one that stands out from others with respect to the substantive strength of a party’s litigating position (considering both the governing law and the facts of the case) or the unreasonable manner in which the case was litigated. District courts may determine whether a case is ‘exceptional’ in the case-by-case exercise of their discretion, considering the totality of the circumstances. (pages 7–8)

What is the ITC?

The International Trade Commission (ITC) is an American administrative body that handles complaints about unfair competition in international trade as defined by law. One type of unfair competition that it can issue administrative rulings against are products made from competitors, imported from abroad for sale in the USA, that infringe one or more patents. One remedy that the ITC can award to successful complainants is a ban on the importation of those infringing products. While it does not have the authority to award damages (ie money), combining an ITC action with a lawsuit seeking damages can be a powerful legal attack against a competitor.

Copyright litigation to watch:

As well as patent action, there’s been some recent copyright litigation affecting open source, particularly Oracle v Google. Oracle sued Google for patent and copyright infringement relating to Google’s use of Java in Android (Oracle acquired the Java copyrights when it bought Sun Microsystems, which created Java in 1995. This litigation has been going on for a couple years now, with an appeals court ruling that Application Programming Interfaces (APIs) can be subject to copyright.

In early October of this year, Google formally applied to the Supreme Court to ask that they hear an appeal on the copyright

issue in this case. It wants the Supreme Court to answer this question: “Whether copyright protection extends to all elements of an original work of computer software, including a system or method of operation, that an author could have written in more than one way.”

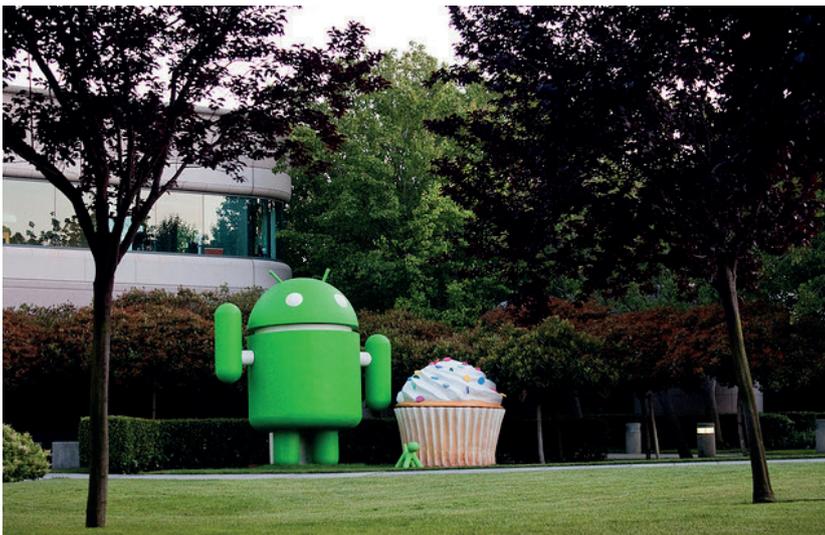
If the Supreme Court hears this case, it’ll affect not only open source software, but the freedom of all software developers working in the United States or for the American market. If the court says that such protection does extend, then we could see sharp limits on third-party developers trying to reimplement APIs of all sorts.

This ruling will weaken patent trolling, as it gives much more leeway to the lower courts to throw out a patent case if they suspect a troll. By making appeals less attractive for trolls, it lowers the chances of them dragging out litigation, making them much less threatening to potential victims.

The Highmark case dealt with how much deference appeals courts should give to district courts that award attorney’s fees in patent infringement cases. Speaking again for a unanimous SCOTUS, Sotomayor ruled, in light of the Octane decision, that there should be considerable deference given to the lower courts: “[Section]285 commits the determination whether a case is ‘exceptional’ to the discretion of the district court [...]” (page 4). This effectively means that a higher court reversing an award of attorney’s fees will become quite uncommon; a significant disincentive to potential patent trolls.



The US Supreme Court may seem remote to non-Americans, but the US is such a huge market that its decisions affect us all. Photo by yeowatzup CC-A 2.0 Generic.



Google also bid on the Nortel Networks patents, but lost out to Rockstar, which bought them for \$4.5 billion (US). Photo: Kenneth Lu, CC-A 2.0 Generic.

Alice v CLS

Alice Corporation, a non-practice patent-holding entity, held patents on a method, system, and process for a particular type of financial risk hedging: namely, that one party to a set of financial transactions won't pay at one or more stages in the set. This risk is known as "settlement risk". Alice's patents describe using a computer to keep track of the transactions between the parties. If the computer determines that a party does not have sufficient funds to pay their obligations to the other side, then the transaction is blocked. Litigation against CLS Bank International for alleged infringement of these patented ideas started in 2007, eventually winding its way up to the Supreme Court of the United States.

Writing for a unanimous court, Supreme Court Justice Clarence Thomas begins with a brief description of what the patents claimed. There are effectively three different types of claims made: "(1) the foregoing method for exchanging obligations (the method claims), (2) a computer system configured to carry out the method for exchanging obligations (the system claims), and (3) a computer-readable medium containing program code for performing the method

of exchanging obligations (the media claims)" (page 3 of the ruling).

Thomas then goes on to cite the court's recent ruling in *Mayo vs Prometheus*, which established a test to determine which inventions incorporating abstract ideas are patent-eligible: "First, we determine whether the claims at issue are directed to one of those patent-ineligible concepts" (page 7). If it is so directed, then the court looks at "the elements of each claim both individually and 'as an ordered combination' to determine whether the additional elements 'transform the nature of the claim' into a patent-eligible application" (page 7). This is what Thomas refers to as "a search for an 'inventive concept'" (page 7).

The court then applies this "Mayo test" to Alice's patents. Beginning with the first step of the test, Thomas notes that these patents do incorporate an abstract concept, as "these claims are drawn to the abstract idea of intermediated settlement" (page 7). Thomas cites previous Supreme Court case law, emphasising its ruling in *Bilski v Kappos* (another case, which threw out a patent on financial risk hedging), as illustrating examples of other patent-ineligible "inventions". Intermediated settlement is an old, abstract financial idea: "Like the risk hedging in *Bilski*, the concept of intermediated settlement is 'a fundamental economic practice long prevalent in our system of commerce'" (page 9).

Thomas then applies the second step: looking at what else is in the patent claims to see if there's anything more to Alice's "invention" than this abstract idea. The method claims don't pass the court's test because they "merely require generic computer implementation" (page 10). In what might be the most important part of the ruling, Thomas emphasises that while some inventions involving computers may be patent-eligible, just because a computer is used as part of an "invention" is not, on its own, enough to get a patent out of an abstract idea (pages 13–14):

There is no dispute that a computer is a tangible system [...] or that many computer-implemented claims are formally addressed to patent-eligible subject matter. But if that were the end of the [Section]101 inquiry, an applicant could claim any principle of the physical or social sciences by reciting a computer system configured to implement the relevant concept. Such a result would make the determination of patent eligibility "depend simply on the draftsman's art" [...]

The remainder of the patents, which are the system and media claims, didn't hold up either. Alice argued that the system claims depend on a particular computer hardware configuration. When the court looked at that configuration, it found only a generic general-purpose computer. Since Alice had stated in an earlier brief that if its method claims fail, so do its media claims, the court threw out the media claims as well. Having demonstrated the court's rationale for invalidating Alice's patents, Thomas conclude the

Post-Alice software patent cases

There have been a number of software patents rejected by lower and appeals courts for invalidity in the wake of the *Alice vs CLS* judgement. Here's a look at three of them:

- 1 **Comcast vs Sprint.** (United States District Court for the District of Delaware) Comcast's patent claim it launched against Sprint Communications described asking a telephone caller if they'd like to make another connection. It was thrown out on the grounds that it is "drawn to the abstract, and fundamental, idea of a conditional decision."
- 2 **Eclipse vs McKinley** (United States District Court for the Central District of California). Eclipse IP owned some patents, one of

which described using computers to communicate automatically with individuals to see if they are completing an assigned task. This was challenged in court by McKinley, a warehouse and loading company and a defendant they sued. The patent was rejected as invalid for being a mere abstract idea.

- 3 **Dietgoal Innovations vs Bravo Media.** (United States District Court for the Southern District of New York) Dietgoal's patent, which described meal planning as implemented by a computer, was rejected as invalid for being an abstract "do-it-on-a-computer" patent and thus "drawn to patent-ineligible subject matter".

ruling by upholding the Court of Appeals for the Federal Circuit's decision to throw out the patents.

Justice Sotomayor also gave a one-paragraph opinion concurring with the court, which was joined by Justice Ruth Ginsburg and Justice Stephen Breyer. A concurring opinion means that some judges agreed with another opinion (in this case, the unanimous opinion of the court), but that they wanted to note another view that they had. In that opinion, Sotomayor notes that the three of them would completely throw out business methods as unpatentable.

This ruling has already had an impact, leading judges to throw out several cases.

NVIDIA vs Samsung *et al*

In September 2014, Nvidia launched two patent infringement actions against Samsung and Qualcomm. Claiming infringements of seven patents, which read on basic GPU functionality, this litigation can't make comfortable reading for those considering opening up their graphics device drivers.

According to Nvidia, it was negotiating with Samsung for a patent licensing agreement. These talks failed after Samsung blamed its hardware suppliers (eg Qualcomm) for infringing, claiming no responsibility themselves. Nvidia then decided to take legal action against both Samsung and Qualcomm.

Nvidia wants money, but it also wants to ban the importation and sale of particular processors and devices. In its complaint to the International Trade Commission (ITC), Nvidia has a non-exhaustive list of Qualcomm products: "These processors include Qualcomm's Snapdragon processors using Adreno GPUs" (page 12). Nvidia also alleges that "Samsung's Exynos processors," which "use Mali GPUs or PowerVR GPUs", found in several of its mobile



US Supreme Court Justice Sonia Sotomayor is one of a number of SCOTUS judges fighting back against patent trolls. Photo by Cknight70 CC-A 2.0 Generic.

What's the difference between patents and copyright?

A patent is a legally granted, time-limited monopoly over the use, manufacture, and distribution of an invention. It's not a right to practice the patented idea, it's a right to exclude others from practicing the invention. While patent laws differ across the world, there are some generalities that we can speak of.

A patent has to be on a new, useful, and non-obvious invention. The idea is that patents are supposed to encourage innovation and reduce reliance on secrecy to the overall benefit of the public. In exchange for a thorough description of the invention to the public (through a patent office), the

patentee gets a commercial monopoly. Some particular ideas can't be patented; in general, things like mathematical formulas or mere abstract concepts on their own.

A copyright is a legally-granted, time-limited near-monopoly over a creative expression. This includes software (in both binary and source code forms), paintings, music, sculptures, and novels. It's a near-monopoly because most jurisdictions offer some form of "fair use" or "fair dealing" exceptions, which let the public use these expressions without having to get a licence. These vary by jurisdiction, but often include uses like parody, news reporting, or satire.

devices, are infringing (page 3). The company also lists Samsung products, which "include, but are not limited to, mobile products such as mobile phones (including the Galaxy Note 4, Galaxy Note Edge, Galaxy S5, Galaxy Note 3, and Galaxy S4) and tablet computers (including the Galaxy Tab S, Galaxy Note Pro, and Galaxy Tab 2)".

There are two separate actions here: a complaint at the International Trade Commission (ITC) and a lawsuit in the US District Court for the District of Delaware. While the ITC can't award damages (ie money), it can stop importation of patent-infringing devices and components. The lawsuit in Delaware also seeks a ban on the allegedly infringing products, but seeks damages too. This double whammy is an increasingly common tactic in the mobile device patent wars.

Those interested in timelines for these processes may not be surprised to hear that they are often drawn-out actions. The ITC decided on 6 October 2014 that it will investigate Nvidia's complaints. This type of investigation could take months or years; we will find out by the end of November when the ITC estimates its investigation will finish. The worst outcome for Qualcomm and Samsung here would be a ban on importing and selling the devices that Nvidia complained about in the United States, which is clearly a lucrative market.

The Delaware action may also be lengthy. We can expect time-consuming motions back and forth from both sides. When you add on to that briefs and memoranda, a pretrial conference, discovery, depositions and oral arguments (not to mention any requests for deadline extensions), a trial could easily be a year away (if it comes to that). The worst outcome for Qualcomm and Samsung here is a US ban on selling the devices Nvidia complained about along with a hefty damages payment to Nvidia.

The impact these proceedings could have might be a further chill on graphics hardware innovation. It may also prove to discourage companies from opening up their graphics device drivers, as they may fear that that would open them up to litigation. ■



Get involved in choosing who benefits from our profits sharing scheme.

It's been 10 months since the first issue of Linux Voice went on sale. So much has happened and it feels like we've barely started. But it's not long before we reach our first major milestone (after hitting our crowdfunding target) – the end of our first year.

We made three promises during our campaign. One was to create the best Linux and Free Software magazine available, and while there's always going to be room for improvement we think we've hit this target. Our

second promise was to release the entire contents of each issue under the terms of the Creative Commons Share-Alike licence and release each issue within nine months of them going on sale. As this very issue circulates the globe, that's exactly what we'll be doing

with issue 1. Head over to LinuxVoice.com and grab a copy, if you haven't already.

Our third promise was to give 50% of our profits back to Free Software and Linux communities. And the best thing about this was that you – our readers – get to choose who gets to benefits. And with only a

couple of months to go before our first year's accounts, now is the time to get started on the process. However, we want the process to be as open and as adaptable

“We promised to give 50% of our profits back to Linux and Free Software communities.”

as possible. Like the magazine, we want to get as close as possible to the ideal solution, but we also want to be open and transparent enough that we can improve the process with your help. Which is why we're starting a little early.

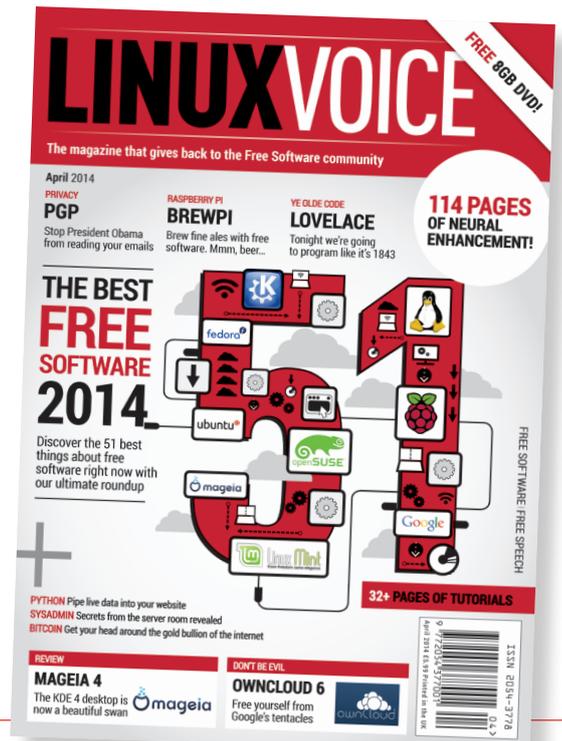
Issue 1 is now yours All of our work for the first issue has been relicensed under the terms of Creative Commons

As per our promise during the Indiegogo campaign that launched this magazine, our first issue, from February 2014, is now available completely free. But just like open source, we don't mean just in the 'free beer' cost sense, but also in the freedom sense. We always felt this was incredibly important for a magazine about free and open source software, and we always wanted to be a magazine that gave back in the same way, with a similar commitment to the same principles that drive open source development.

This is also the reason why we chose the Attribution-ShareAlike variant of the Creative Commons licences. This allows you to take our content and copy and redistribute on any medium or in any format. Just like open source, you can also edit, transform and build on any of our work for any purpose – even commercially! The only conditions are that you need to give us credit (attribution) and document any changes that you've made. And also like open source, you need to make those changes available under the same terms.

One example of what can be done with the content is that a few of us here on the magazine and many from the Linux Voice community have taken those words and turned them into a freely distributable audio recording. We've split the recordings by article and also made a complete RSS file if you wanted to grab the long play version. It's not bad for a first effort and it's brilliant hearing everyone's contribution. We're all extremely grateful for the support from the various people who have made their own recordings (you only need a microphone and a copy of Audacity if you want to help with our next issue). Not only does the audio version make our content more accessible, we also think that the spoken version makes an excellent time killer for long journeys and train delays. Take a look at LinuxVoice.com for more information.

Download, modify, share and upload our very first issue, thanks to the Creative Commons.



Central to our strategy are the 114 comments we received from our blog post "Giving Profits Back: Where and How", which we put online in November 2013. As you might expect when you ask a question like this, there are many different points of view, many of which may be incompatible with one another, or even have the potential to cause conflict. This is the absolute last thing we want to happen from doing this, so we want to tread carefully. At least for our first year, as we expect profits to be modest, and we can fine-tune the process and set a firm foundation for future Linux Voice awards.

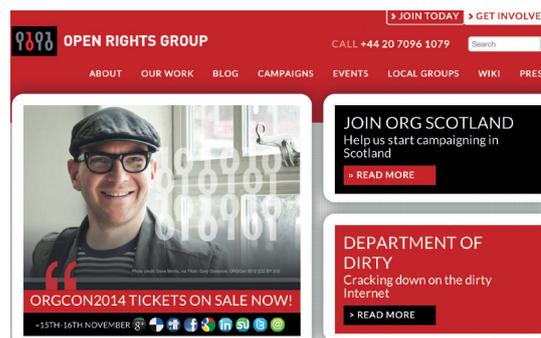
Apart from the many specific projects, communities and foundations that were mentioned in that original post, there was a central point that was repeated by many: both large and small projects should benefit. This was primarily because most of us agree that large projects, such as the Free Software Foundation or the Electronic Freedom Foundation, should still be supported, but so too should the smaller projects, perhaps to help with web hosting, or to help a developer attend a conference. That means we've got to split the overall contributions into different categories, and for that, we're going to need a shortlist of both categories and the projects within them.

Voting process

We've got our own strong ideas about the kinds of projects and communities we'd like to support. We've spoken about and spoken to many of the people and communities we'd like to see, and we've covered a handful of our favourites over the last 10 months. But it's vitally important that our readers get to choose not only who wins any awards, but also what projects go

into the selection pool. For that reason, we want to split the process into three stages. The first stage is going to be open to everyone who may want to participate, and we'll use this stage to formulate a shortlist of projects that go into the second stage. The second stage will be the voting process where our readers will decide on which projects they'd like to see succeed, and the final stage will be the awards themselves.

The list of potential projects out of those mentioned in the "Giving Profits Back" story comments is huge, mostly split between what we'd call organisations (24 from the original post, including FSF, EFF, Mozilla, etc.), distributions (nine in the post, including Debian, Slackware, Crunchbang) and an enormous variety of individual software projects. We thought about splitting this huge category into sub-categories – maybe one for graphical applications, another for APIs or frameworks and another for desktop environments, for example, but we're guessing this would get complicated quickly and maybe add a little unnecessary conflict when people argue over our



The Open Rights Group has similar aims to the EFF but UK-based and focused on UK activism, including fighting the Digital Economy Act and the default use of content filters. www.openrightsgroup.org

arbitrary categorisations. But it does seem like we could make a sensible split by putting the organisations and the distros into one pool, and the huge variety of software projects into another. This would allow us to reward more projects, and reward both big and small at the same time – depending on the final vote of course.

With two categories – one for organisations (where we include distros), and another for software projects, we now want to create two smaller shortlists that can be put to a final vote. The reduction to a smaller number is because we don't want an unmanageable pool of proposals, and to give everyone a better focus on the final projects that have contention. As this issue goes out, we'll ask again what organisations, distros, software, projects and developers people would like to see supported, and depending on the

balance of proposals, we'll put the most popular 10 proposals for each into the two categories, creating those two shortlists. We'll do this publicly so that everyone can see the progress of their favourite projects until we have a final list at some pre-determined cut-off point.

We're going to restrict voting to just our subscribers. This is because it will be a lot easier for us to manage – we'll be able to easily add the voting interface to the online subscriptions page for each subscriber, for example, and this limitation also recognises our subscribers for the contributions they're making. Our plan is to then have three different awards for each category – one main 'winner' and two 'runners up' – making six awards in total, which we'll announce after we've hired an accountant and we know how much we have to give away.

Suggested projects

A quick overview of many of the projects, distros and software proposed online when we asked which our readers would like to support.

- **Accessible Computing Foundation** Making technology accessible to people suffering from various disabilities that normally make conventional computing extremely difficult, if not impossible. <http://accessiblecomputingfoundation.org>
- **Document Foundation** Primarily supports LibreOffice.org in attempting to give everyone access to office productivity tools free of charge. www.documentfoundation.org
- **EFF** Defending our civil liberties in the digital age, the Electronic Frontier Foundation champions privacy, free expression and innovation. www.eff.org
- **FSF** Promoting computer user freedom and the rights of free software users, the FSF is also the custodian of the GNU project and licences. www.fsf.org
- **FSFE** Sharing all the same principles as the FSF, the FSFE works with the European Commission and Parliament to create a positive environment for Free Software and Open Standards. <http://fsfe.org>
- **Gnome Foundation** Furthers the goals of the Gnome project by coordinating releases, sponsoring GUADEC and liaising with both commercial and non commercial organisations interested in Gnome. www.gnome.org
- **Hacker Public Radio** Releases a new community produced podcast Monday – Friday with no restrictions on length or content as long as the subject is likely to be of interest to "hackers." <http://hackerpublicradio.org>
- **KDE eV** This is the non-profit that represents the KDE desktop and the KDE community securing cash, hardware, and other donations to aid development and promotion. <https://ev.kde.org>
- **OSI** The Open Source Initiative is a global non-profit curating and promoting the definition of open source and the licences that fit its description. <http://opensource.org>
- **Outreach Programme for Women** Annual internships to help women get involved in Free and Open Source software. <http://gnome.org/opw>
- **Practical Action** Using technology to challenge poverty in developing countries by building skills and knowledge. <http://practicalaction.org>
- **Software Freedom Conservancy** Promotes, improves and frequently defends Free and Open Source software. Projects under its wing include *Git*, *Inkscape*, *phpMyAdmin*, *Samba*, *SugarLabs*, *Wine* and many more. <https://sfconservancy.org>
- **Young Rewired State** An independent global network, based in the UK, that helps kids ages 18 and under to use open data to make websites, apps and algorithms for solving real-world challenges. <https://youngrewiredstate.org>

The Free Software Foundation Europe is a separate organisation from the FSF, and doesn't get the credit it deserves.

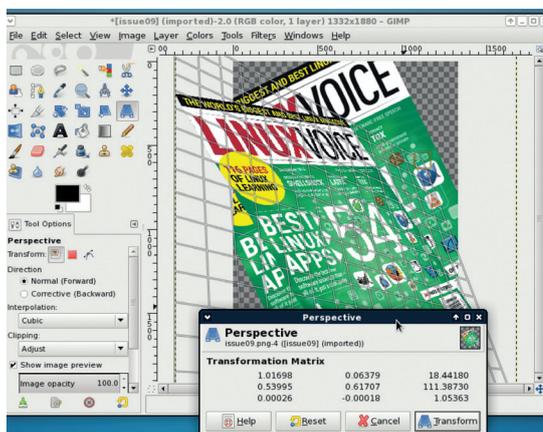


Distributions

- **Crunchbang** A Debian-based distribution with a great blend of “speed, style and substance” thanks to its use of the Openbox window manager <http://crunchbang.org>
- **Debian** Announced in 1993 with a first release in 1996, Debian is one of the pillars of the GNU/Linux distro community thanks to its emphasis on community and stability <http://debian.org>
- **FreedomBox** A private server that integrates strong privacy and encryption to deliver many of the same features you find from online cloud servers and services <http://freedomboxfoundation.org>
- **Linux Mint** One of the most popular distributions using either Ubuntu or Debian as a back-end and responsible for the development of the Cinnamon desktop and the Gnome 2.x fork, Mate. www.linuxmint.com
- **Manjaro** Based on the newbie-impenetrable Arch and using the Xfce desktop, this is a rolling release distro with an emphasis on ease of use and installation <http://manjaro.org>
- **Replicant** A fully open source version of Google’s Android operating for certain specific mobile devices system that promoted freedom, privacy and security issues www.replicant.us
- **Slackware** The oldest currently maintained Linux distribution, Slackware is still a benchmark for simplicity, security and stability www.slackware.com
- **Sonar GNU/Linux** An accessible distribution that bundles assistive technology like a screen reader, magnification, on-screen keyboard and a dyslexic-friendly font <http://sonargnulinux.com>
- **Trisquel GNU/Linux** Based on Ubuntu but with a commitment to remove all proprietary and non-free elements, it’s one of the few distros recognised by the FSF as containing only free software.

Software

- **Audacity** As used by podcasters the world over. cross-platform audio waveform editing unrivalled by functionality or price. <http://audacity.sourceforge.net>



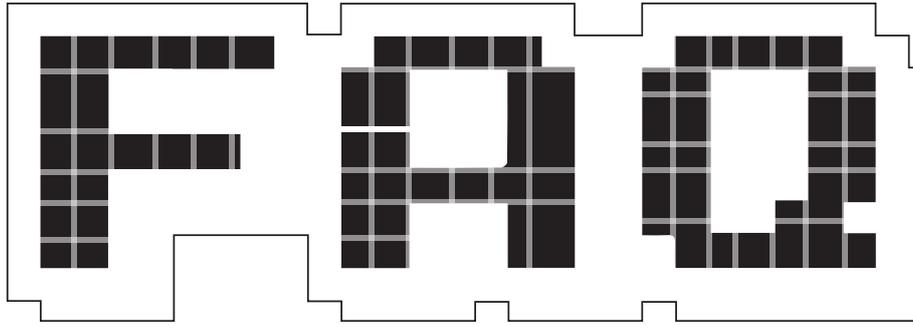
A few readers reckon *Gimp* could do with a bit of cash.



Organisations such as Young Rewired State are doing a fine job of plugging the gap in the UK’s IT curriculum.

- **Blender** A 3D sculpting, animation, rendering and games engine that rivals software costing thousands and is quickly becoming an industry standard. www.blender.org
- **Gpodder** A popular choice for listeners to our own podcast. It downloads and plays new episodes automatically. <http://gpodder.org>
- **Inkscape** A wonderful vector drawing application that’s capable of brilliant professional results and even better output. <https://inkscape.org>
- **MediaGoblin**: A decentralised media publishing platform trying to compete with YouTube, SoundCloud and Flickr. <http://mediagoblin.org>
- **Meld** Compare text/code files and visually spot the differences between them. Great for deciphering drunken coding sessions. <http://meldmerge.org>
- **Nouveau** These drivers give Nvidia owners graphics acceleration without resorting to closed proprietary drivers. <http://nouveau.freedesktop.org/wiki>
- **OpenSSH** This secure remote shell is as essential as oxygen to anyone using Linux on a remote device. www.openssh.com
- **Scribus** Desktop publishing that can take your words and images and turn them into something as magical as a printed publication. www.scribus.net
- **VLC Media Player**: It plays everything, streams everything and runs on almost everything. It’s the media player most of us rely on the most. www.videolan.org
- **Xmonad**: Tiling window managers make you look cool. Yes. But they also make you incredibly productive by keeping distractions and mouse clicks at bay. <http://xmonad.org>
- **ZoneMinder**: Home security is normally expensive, which is why this fantastic open source solution starting with just a webcam is such a great project. <http://www.zoneminder.com>

Now it’s up to you – have a think about which projects you’d like us to support, and keep an eye on LinuxVoice.com for the revised shortlist.



OPENRISC

Open source software is all good and well – but how about open source CPUs?

MIKE SAUNDERS

Q So, this is free software clone of a certain famous board game, with some kind of pun in the title, right?

A No, this has nothing to do with Risk. It's all about CPU cores here – and specifically, RISC (reduced instruction set computing) ones. OpenRISC is a project that designs completely open processors, which you can study, modify, and have produced in factories. Well, providing you have enough money to do that, of course...

Q Hang on a minute! Reduced instruction set? Why would I want that? Give me more, more, more instructions!

A Actually, RISC designs go back a long way, and have nothing to do with the overall power of a CPU. They're all about designing the processor so that it has fewer jobs to do – but it can do them more quickly and efficiently than in other designs. Historically, x86 CPUs have been complicated beasts,

“RISC designs go back a long way, and have nothing to do with a CPU's overall power.”

with each generation adding more and more layers of cruft onto the original design. You end up with some CPU instructions taking just a few clock cycles to execute, whereas others take far longer.

RISC, however, opts for a much smaller range of instructions (and therefore fewer transistors on the chips), and these instructions are highly optimised. Fewer instructions doesn't mean less capability though; after all, CPUs just move numbers around in memory and perform calculations on them. And today, RISC has won: ARM chips are built with RISC architecture and are absolutely everywhere, and even though x86 CPUs are still regarded as complex (Complex, rather than Reduced Instruction Set Computing) in their architectures, modern ones break instructions down into smaller components, in a RISC-like fashion.

Q OK, but CPUs are hardware – how do you make them open source? And why would you want to?

A Yes, hardware doesn't grow on trees (unless you have some fancy wood panelling for your PC case), so initially it might seem odd to apply FOSS principles to it. But consider designs: can you take your current PC or laptop processor, get a complete specification for it, change a few parts and make your own version? Unless

you're willing to pay giant licensing fees to Intel or AMD, this probably isn't an option. And even then, you wouldn't necessarily get to share your changes with the rest of the world.

Now, imagine if the entire design of your CPU were open. Imagine if the community could work together on improving its performance, features and power management. Imagine if this could work in tandem with an entirely free software stack, so that every byte and electron of your computer was free as in speech.

Q That sounds like heaven for Richard Stallman, but how many geeks really want to fiddle with CPU designs?

A Just because it seems like a black art, or an obscure subject, it doesn't mean that nobody is interested in it. On the contrary – at the time of writing, the third annual OpenRISC Conference (ORCONF 2014) was taking place in Munich, Germany. Forty developers from around the globe gathered together to discuss the current state of OpenRISC, share projects that they're working on, and contemplate ideas for the future – see page 16 for a full report.

Q So, how are OpenRISC chips made? How is development done, and who produces them?

A OpenRISC came to life in 1999, the work of a few computer science students in Slovenia, and received a small amount of financial backing in the early 2000s. Today, the OpenRISC project is much bigger and has a few chip designs (known as “cores”), but currently the focus is on the OpenRISC 1000, also known as OR1K: <http://opencores.org/or1k>. This CPU is focused on networking and embedded tasks, with special emphasis placed on simplicity and low power consumption. Its design is constructed using the Verilog hardware description language – and indeed, you can see the source code for yourself at <https://github.com/openrisc/mor1kx/tree/master/rtl/verilog>. This code is released under the GNU Lesser General Public Licence, so you’re free to base your own chip on it, as long as you make your designs available for everyone else.

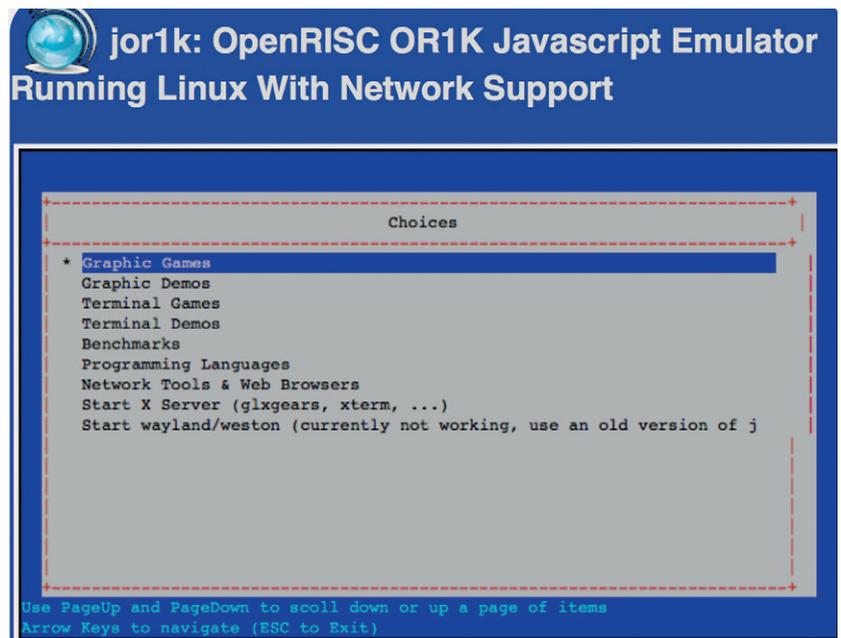
For development purposes, the OR1K design can be implemented in FPGA (field-programmable gate array) development boards, which means you can hook up an OpenRISC processor to various peripherals and create a fully working computer. The GNU toolchain has been ported so that it can produce OpenRISC binaries, and work is underway to get *LLVM/Clang* to the same state as well. The Linux kernel itself has been able to run on OpenRISC since version 3.1, and a few other real-time operating systems such as eCos have OpenRISC ports as well.

OpenRISC is popular in embedded devices, and at ORCONF one developer talked about a project using OpenRISC chips in TV set-top boxes to convert satellite video data to internet streaming formats. Other embedded chips had been considered, but it was important for the set-top box manufacturer that the chips could run the Linux kernel, so OpenRISC was chosen.

Another area in which OpenRISC is well known is academia. The Technical University of Munich uses it in research (which is why the conference was held in the city), while many other universities teach courses based on it.

Q **What about the chip in the real world, though – are there any commercial implementations?**

A You bet! This isn’t just a play toy for hackers and students.



Experiment with OpenRISC in your browser: try the JavaScript-based emulator (running a Linux kernel) at <http://s-macke.github.io/jor1k>.

OpenRISC has been implemented in many SoC (system on a chip) designs – that is, chips that include all the components needed for a standalone computer. Most notably, Samsung has used it in various digital TV models, while the Allwinner A31, a SoC used in a large range of mobile phones, tablets and smart TVs, has an OpenRISC core doing power management work.

OpenRISC has even gone into space as part of the TechEdSat project, which is a satellite designed by students at San Jose State University – it needed to be cheap, so OpenRISC was a natural choice. It could be used in many other devices and projects as well, that we don’t even know about, just as Linux and the BSDs are often chugging away inside networking and embedded hardware, without any fanfare.

Q **Is OpenRISC finished, or is it still a work in progress?**

A With 80,000 lines of Verilog behind the design, there’s still plenty of room for tweaks, performance improvements and power savings. A team of developers is beavering away on a new CPU pipeline – that is, the series of stages used to process instructions. Ideally, it will support out-of-order execution, along with dynamic branch prediction. Another topic that came up at the conference was improving the 64-bit version of the

chip. There’s even some interest in porting FreeBSD to OpenRISC, although we’d wager that NetBSD will get there first, given that it already runs on a staggering number of platforms...

Q **OK, you’ve sufficiently whetted my appetite, and now I want to make big bucks as a CPU designer. Where do I start?**

A Well, we can’t guarantee that you’ll be the Chief CPU Architect at Intel after playing around with OpenRISC for a few months, but it’s a great way to dip your toes into the vast world of processor design. A good place to begin is the Getting Started guide at <http://kevinmehall.net/openrisc/guide> – and note the requirements in particular. You should have a solid grounding in the workings of a Linux installation, along with knowledge of Verilog and C (the language used to code the Linux kernel).

The guide also explains how to run *or1ksim*, an OpenRISC CPU emulator, and how to use the chip on an FPGA development board. You can pick up one of these boards for around £50 to £100; see http://opencores.org/or1k/FPGA_Development_Boards for a list of boards that work with OpenRISC. But if you really want to dive in at the deep end, try the Architecture Manual at <http://tinyurl.com/qj6pjfc> – just bear in mind that it’s 358 pages long! 

< TIM BRAY />

Graham Morrison meets the co-author of XML, a fierce privacy advocate, an encryption hacker and the owner of many hats.

Watching Tim Bray talk to an audience is a little intimidating. He talks fast and every word counts. And he wants action – he wants his audience to change the world. After founding companies, co-authoring the XML specification, working at Sun Microsystems and then Google (leaving because he famously didn't want to leave Canada for Silicon Valley), Tim has seen, thought and

talked about most things to do with technology. He's even making his own security contributions to the amazing open source Android email application, K-9. His keynote at OSCON 2014 was about threats – threats to our privacy, threats to our online freedoms and threats to our data, and "Now is the time for sensible, reasonable, extreme paranoia," as he puts it. Which is exactly what we wanted to talk about when we met with him.

LV How do we balance the advantages of big data with healthcare systems and privacy? And how can PGP help with this?

Tim Bray: Well Larry (Wall, the creator of the Perl programming language) certainly puts things into sharp perspective. You know, if you're willing to trust people with your health data maybe we can save a lot of lives, and yet there are a substantial number of people who are reasonably paranoid about trusting their data to anybody. So if you're going to do that, you should be transparent about how you're going to handle the data, how you'd protect it and who you'd share it with, and so on.

But I don't think that's anything like a complete answer, because the vast majority of people don't really have the education or tools to understand what the data protection options are, and what the consequences of sharing are. And we shouldn't expect them to. It's a hard area of expertise and not one that we should expect civilians to have. So I think there's actually quite a strong role for the public sector in here to establish regulations, because this is not an issue

of technology, it's purely an issue of policy. Now in the US, they have such regulations, the most visible of which is called HIPAA, which is widely used and strictly enforced in the healthcare sector. And, whereas I think almost anybody would agree that it's reasonable to have such regulations, the HIPAA itself is not actually very good. In particular, it does things like mandate the use of specific technologies such as passwords when, quite possibly, better alternatives to passwords would be more secure and less onerous are available.

So I would probably appeal to a combination of market forces and public regulation to assume the default action is the right action, is the ethical action, is the action that produces the least surprise on users. If we've learnt one thing, no matter how many information popups you put there saying what you're going to do with the information and what the consequences are, a high proportion of people will say 'ah, where's the OK button, I just wanna get through this'. And whether you like it or not, that's just



the way people are. So I think that we, as a profession, have a responsibility to do the ethical thing by default and not surprise people with what happens to the information they share. As you point out, OpenPGP is a well established, fully debugged cryptography framework that is highly implementable and has been implemented lots of times, and yet its penetration among the non-geek community is more or less zero percent. If we're not going to make it easy to use by default, then we're not going to have privacy by default and I think our profession owes the community, the people that use our stuff, privacy by default.

LV If you look at HTTPS (ignoring Heartbleed), even people without technical knowledge can



“We owe Mr Snowden a lot of credit for exposing people to the fact that their governments are doing things of questionable legality, morality and effectiveness.”

understand that a site is encrypted, at least in principle. The certificate sharing is all done transparently and there’s an authority that deals with the authentication of the certificate. Maybe that’s what we need?

TB: Whenever anyone mentions HTTPS, flawed though it is, I have to push back a little bit. Yeah, it’s possible to screw up the installation and deployment and so on, so that it doesn’t work properly. But when it is deployed properly, which is not that hard, there is not a known instance of HTTPS data

“We, as a profession, have a responsibility to do the ethical thing by default.”

being brute-force decrypted. It is plenty good enough. And yes I agree with you that people are starting to be educated about what the little lock means.

One of the things that I’m disappointed about though is the community of developers that are still offering services over HTTP without even allowing HTTPS let alone requiring HTTPS. What they should all do is go to their legal department and ask a legal opinion, ‘Hey, we’re offering our services in such a way that anyone sent onto the company can spy on us – is that legally OK?’ Well, you know what, it very probably isn’t!

LV **Even on Android, we don’t know of an email client with decent S/MIME PGP support.**

TB: So K-9 is getting it.

LV **But it’s taken a couple of years to get it into the latest alpha.**

TB: Well, one of the problems with cryptography is key management, and how do you find the key for the person you want to send a secure message to. Historically, the cypherpunk community who invented all this stuff were a little bit guilty of letting the perfect be the enemy of the good, and so they set up this web of trust mechanism, which never worked – ordinary people won’t go to key signing parties. I’m sorry, it’s not going to happen.

So I’ve been super interested in this thing called keybase.io, which provides a pretty nice solution to the problem of acquiring a key with good reasons to believe in it. And, what’s great about it is that you don’t even have to trust keybase.io. They present everything



"I don't care if Apple has signed it, I want to download an app that has previously been audited and found to be safe."

that you need to know in proofs that you can independently verify if you want to. I thought the difficulty of actually discovering and acquiring keys might be the single largest remaining logjam, or one of them anyhow, in making quality cryptography available. Now I think we're ready to make that one go away, and so I'm optimistic.

LV GnuPG for email clients can work completely transparently as far as key exchange and key discovery go. You don't even have to know anything about the keys.

TB: And particularly in the case of keybase, you just go 'I'm looking for a key for you', and it says 'Oh, here's a key, which is also controlled by the person who controls your Twitter account and your GitHub account'. OK, that's good evidence, I'll believe that that key is a good one to encrypt my message with.

LV When do you think this became such a big issue? Has it always been an issue, or has it become more an issue as the internet has become so much a part of our lives?

TB: I would offer a lot of credit to Ed Snowden, who started this conversation. I think a lot of people, security professionals and people close to metal, kind of knew what was going on. Knew that, you know, government agencies were in fact scooping up a

huge proportion of internet traffic. We did not know they were tapping into Google's data centre.

LV And you worked at Google for a while...

TB: Yeah, and I tell you, the people at Google were livid when that one came to the surface. So I think we owe Mr Snowden a lot of credit for exposing the larger population to the fact that their governments are doing things that are of questionable legality, morality and effectiveness.

LV In the UK – thanks to the European Court of Justice – it is illegal to harvest data from ISPs on a large scale and that's why they're rushing through these very quick surveillance laws to make it 'not' illegal.

TB: Right, the DRIP [Data Retention and Investigatory Powers] law. We're having a similar problem in Canada, although one of our courts found that it is illegal for ISPs to provide usage data without a warrant, which is a change in the landscape.

LV Do you think it's too late? Do you think the cat's out of the bag?

TB: Oh no, not in the slightest. We're still in the first generation of people who live on the internet. We can make the

experience more private and safe for all the people that are going to come after us.

LV But even with all the data anonymised, there's no one accepting anyone's word that it's anonymised, and even when it is anonymised, you only need something like three trig points to find people. There's just so much data out there. It seems to us like the cat's already out the bag.

TB: Well sure, you can take that attitude. And also you can say that anyone who carries a mobile phone is revealing their location at all times. But I think the notion that the cat's out the bag and we should all roll over and give up is just barshit. I think that every time we increase the proportion of the traffic that is privacy-enabled, we are doing a public service. We are driving up the cost to those who have been abusing the existing transparency, and I think that's something we should do. We have no excuse for not doing it.

LV But we're in the minority. The new generation are used to Facebook and everything about their lives being online.

TB: From what I hear, the younger generation is increasingly less Facebook-centric and one of the reasons for that is they don't like that

creepy feeling that their information is leaking. Just because an opinion one holds is in the minority, doesn't mean we should give up and stop pursuing it. And, like I said earlier, I don't think we should expect the general population to have educated opinions about internet safety any more than they have educated opinions about emission control systems in automobiles or air traffic control regulations, and that kind of thing. It's our responsibility as professionals to ascertain what the most beneficial behaviour is and build things in that way.

“I'm not arguing in favour of absolute privacy; there can never be such a thing.”

LV It's one thing to say it, but we've known about this for a long time and it hasn't happened.

TB: I'm disturbed by your counsels of despair. The proportion of things being transmitted over HTTPS is monotonically increasing. And we still encounter people saying 'oh, my stuff is shareware, it doesn't need to be HTTPS'. And to that I say, 'well, yeah, but the decision as to whether something needs to be private is complicated and very context dependent, and most people aren't

equipped to make it, so why don't you just do the safe thing and make everything private. And that's an unanswerable argument now that HTTPS is very low cost and low difficulty to deploy. I'm quite optimistic actually. I'm not arguing in favour of absolute privacy; there can never be such a thing. All we can ever do is move the needle and increase the proportion of things that are privacy enabled, and I think we're doing that.

I notice that there are excellent *OpenPGP* libraries available now for Ruby and Python and JavaScript and Node. There are not for Java itself or for .Net, which is a problem that we need to address. I know that people are working hard over at Google and some other places about trying to do crypto in the browser. Which is, if you can do it, a super interesting enabler of a bunch of interesting things, but there's a severe question of trust. When can you trust an app that you download over the air? Those problems are not insuperable.

LV Do you think we can counteract the new app-centric frontier by using stuff like Firefox OS?

TB: It is clearly the case that the browser is becoming less central as part of the software delivery ecosystem because a very high proportion of things are migrating into native mobile apps. And to some extent that's

happening for reasons that are actually good. Native mobile apps offer more responsiveness and a lot of UX options that are not available in browser apps. They also have an important advantage that we're not giving enough credit to, which is that they also have a superior developer experience. However, there are two important advantages of the browser-centric world that we're losing. One of course is the speed of update. If you have a bug that you need to fix fast, in the case of Android you're looking at a latency of hours and in the case of Apple you're looking at a totally unpredictable latency of potentially weeks, whereas if you're web-centric you can fix an applet in minutes. And that's huge. Secondly, the issue you raised is important to me. I think the internet is better when anyone can deploy anything on it without asking for permission. And anyone can go and get it. And I do not like the app store as intermediaries through which you go to find anything. I'm less offended by the Google app store than by Apple's because of its exclusivity – when Apple says we're not going to sell something, they're not saying they're not going to sell it, they're saying you can't have it. If Google doesn't like something, well you can still get it off the project's website, right? How much impact is something like Firefox OS gonna have? Well, I hate to say it, but that probably depends on price performance of the hardware it's running on.

LV We lament the loss of what the internet had at the beginning of its life.

TB: Think about it this way, the volume of traffic on the internet is unimaginably vast. So whether you are an over enthusiastic government agency or whether you are a hacker trying to steal credit card numbers, every time we can impose a non-zero cost on surveillance, the volume is so high that there will be entire class of surveillance or an entire class of crime, that it becomes uneconomic. And, given that, why would we not do it? Every time you increase the proportion of traffic with HTTPS up by 5%, you drive a certain class of undesirable behaviours into uneconomic territory. You don't have to win to win, you just have to make things better to improve the environment. **LV**



Tim's a fan of the way apps make things easier for developers – but not of the disadvantages they present to users.

Write once,
deploy everywhere.



ubuntu[®]

BUY LINUXVOICE MUGS AND T-SHIRTS!



shop.linuxvoice.com

LINUX VOICE REVIEWS



Andrew Gregory

So this is what bandwidth shaping looks like. Thanks BT, thanks a lot.

One of the few good things to have come out of the recent/current Gamergate campaign of harassment is that Kathy Sierra has returned to the internet, this time to introduce (to me, at least) the concept of the Kool-Aid point. This imaginary construct is that point at which something achieves a degree of popularity; trolls are then given licence to attack the idea/person/thing merely because it's popular. The merits of the thing in question go out the window; all that matters is that if it's popular, it is by definition inferior in some way.

Ubuntu is popular. It's still the Linux distro with the biggest chance of getting through to the masses, and there's a predictable queue of critics lining up to slate it for any number of reasons, both real and imagined. Likewise Google. Google is an advertising company. Its key revenue driver is advertising, built off the back of the best search engine in town. It doesn't need to make 3D cheaper for everyone, or provide free translation services, or do any of the other things it does that make our lives easier. By all means knock them for the creepy privacy invasions, but Canonical and Google aren't privatised state monopolies like BT – we can't knock them just for being successful.

andrew@linuxvoice.com

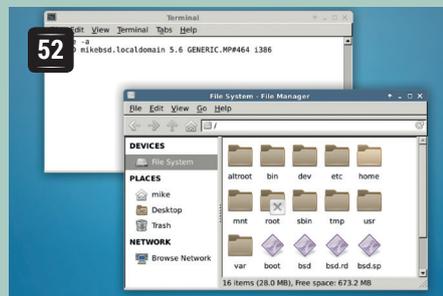
The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

On test this issue...



Ubuntu 14.10

Ben Everard is bemused by the newbie distro *par excellence* offering upgraded developer tools in its latest offering.



FreeBSD 5.6

The grass may not always be greener, but **Mike Saunders** does like to look at an alternative OS to Linux now and then.



FRITZ!box 7490

Paranoid BT customer **Graham Morrison** switches router to a device that won't send all his data to GCHQ.

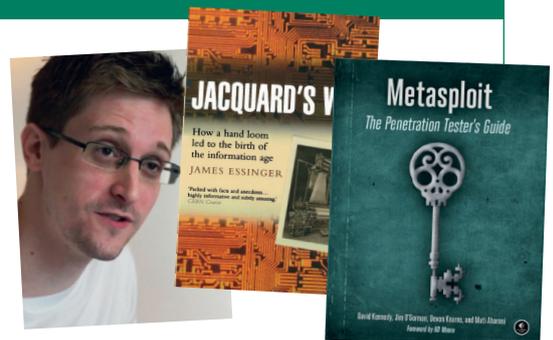


Google cardboard

After a greasy pizza, **Ben Everard** sticks the box on his face and pretends he's in *The Lawnmower Man*. Thanks Google!

BOOKS AND GROUP TEST

One of the best things about Linux from an ethical point of view is that it runs on old hardware. Linux and free software has the potential to keep tonnes of rare metals out of the landfill site (and save you a few quid in the process), and it's all thanks to lightweight Linux distributions such as those tested on page 56. If you have a machine gathering dust, install one of these and marvel at the resurrection. And over in book reviews there's a filmic interloper in the shape of patriot/traitor Edward Snowden biopic *Citizen Four*. Seek it out; it's rather good.



Ubuntu Utopic Unicorn (aka 14.10)

Long-term Ubuntu user **Ben Everard** discovers whether Deckard was a replicant.

DATA

Web
www.ubuntu.com
Developer
Canonical
Price
Free under various
licences

The biggest news about the Utopic Unicorn isn't what's included in it, but what isn't. Mir (a display server) and *Systemd* (an init system – see page 86 for much more) are both coming to Ubuntu, and they'll both have a profound effect on how it works. However, they're both going to arrive in the release after this (15.04, aka Vivid Vervet). Without these, Utopic Unicorn is a boring release in many ways. Without Mir, there's no Unity 8, so the desktop environment hasn't changed significantly from 14.04 (though it does work better on High DPI displays)

In fact, as far as desktop users are concerned, the biggest changes are the inclusion of the latest packages.

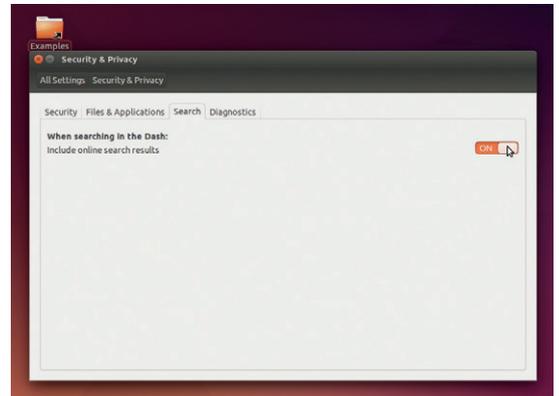
- **Linux 3.16** The kernel upgrade brings some performance improvements.
- **Firefox 33** Brings some significant performance improvements to web browsing.
- **LibreOffice 4.3** This has improved DOCX support

Other than this, the updated versions of **Libnss** means that Netflix will now work out of the box with Chrome. Unlike the previous release (Trusty Tahir aka 14.04), Utopic is not an LTS (Long Term Support)

release. This means that it will only be supported for nine months. Since releases come out every six months, this

“Anyone wanting to develop on cloud services may find that Utopic is the best distro for them.”

means that users either have to upgrade every time, use an unsupported system or switch to another distro altogether. Usually, upgrading each time isn't



Unity will still include online results in searches in the Dash – a move that angers many privacy campaigners.

a big problem, but if both *Systemd* and Mir arrive on schedule, they'll both be coming in 15.04. This means that by choosing to install 14.10, you are effectively choosing to switch to *Systemd* and Mir in a few months' time. Even if you like the idea of these technologies, the first version of a distro containing them is likely to have a few problems. Unless you're prepared to endure these problems, a more prudent approach would be to install 14.04, which will be supported until April 2019.

Developers developers developers!

There is, really, only one significant release from Canonical in 14.10, and that's the *Ubuntu Developer Tools Centre (UDTC)*. The idea behind this tool is that it will enable developers to quickly set up common development environments. The first environment is for Android development including the various tools from Google. It's not hard to set these up, but there are a few bits that need to be downloaded, and dependencies installed. *UDTC* acts like a sort of package manager that enables you download everything and set it up in one go.

This is useful – it slightly reduces the amount of work that a small minority of people will have to do after installing Ubuntu – but it's not really a killer feature. After all, not many people will even need it. We can certainly see it becoming more useful over time. For example, if it becomes easy to define the environments that get installed, you could create custom ones for particular projects so the whole team can easily stay updated, and new members could instantly get everything they need. However *UDTC* is still some way off this ideal.

If you're using Ubuntu 14.04 and like the sound of the *UDTC*, you don't need to upgrade your whole

Flavours

Of course, Ubuntu isn't just about the default version with the Unity desktop: there are seven official flavours of Ubuntu with different desktop environments. Canonical hasn't released any information about how popular they are, so we've ordered them by the number of people seeding the ISO on BitTorrent.

- **Ubuntu GNOME** The newest official flavour of Ubuntu has proved to be very popular, and shows that many Ubuntu users would like to stick with the distro's roots. Unfortunately, the release schedules of Ubuntu and GNOME just fell out of sync, so 14.10 ships with GNOME 3.12 instead of the slightly newer 3.14.
- **Kubuntu** KDE is also in for a big upgrade soon. By default, Kubuntu still ships with Plasma 4, but intrepid users can switch to

Plasma 5 and experience the future.

- **Xubuntu** Not much has changed in Xfce, but Xubuntu users can upgrade to take advantage of the latest packages.
- **Lubuntu LXDE** only gets some bugfixes this version, as most of the development is going into LXQt. This is another distro with a big change coming up.
- **Ubuntu Kylin** The Chinese version of Ubuntu comes with a more tweaks than just the language. For example, *WPS Office* ships as the office suite because it's more popular than *LibreOffice* in China.
- **Ubuntu Studio** As well as the usual raft of changes, the new kernel brings ALSA support for firewire devices.
- **Mythbuntu** This hasn't released a 14.10 version as there weren't enough changes to make the non-LTS release worthwhile.

Desktop next

As we've mentioned, 14.10 is, well, a bit of a boring release for the desktop. But this doesn't leave the thrill-seeking distrohopper without an exciting version of Ubuntu. As we've mentioned, there are a lot of changes stacking up for the next version (15.04, aka Vivid Vervet), and there are already images of this distro that you can install.

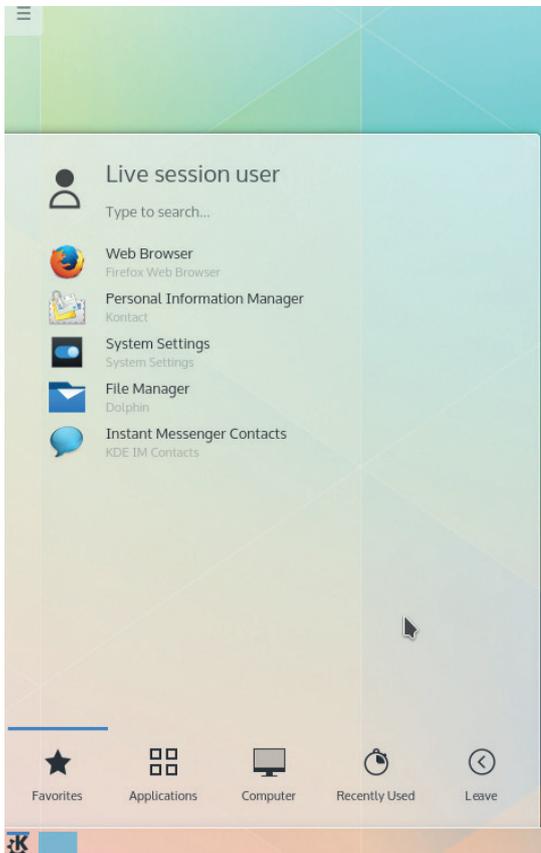
Given that this is still months away from release, you should expect quite a lot to be broken, and even more to break with upgrades, so it probably isn't suitable for a main computer yet even if you do know what you're doing. However, if you want to see what's going to happen next release, fire it up and have a look. In some versions of *Virtualbox*, it's not starting correctly. If you end up with a screen of colourful characters instead of a desktop, press right-Ctrl+F1, then right-Ctrl+F7.

distro; you can get it by adding the PPA:

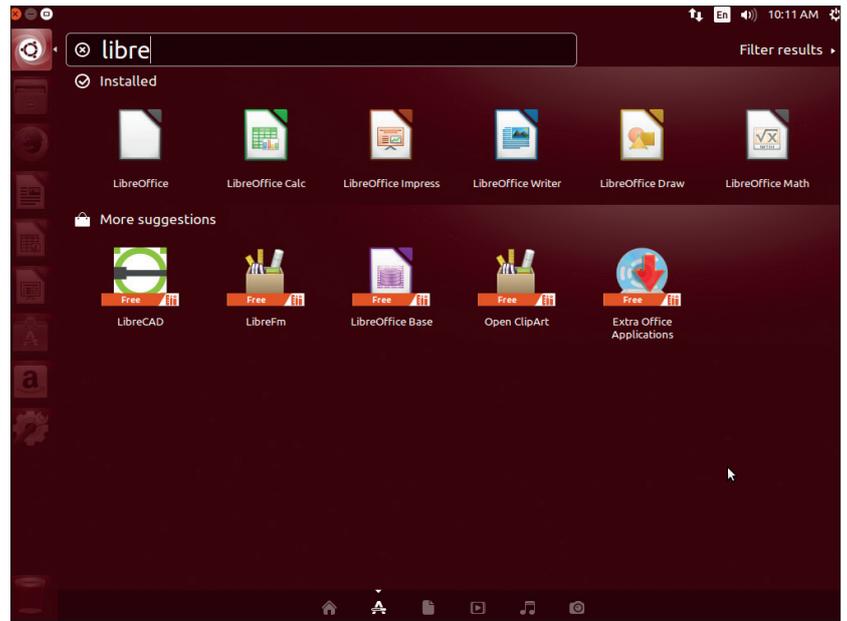
```
sudo add-apt-repository ppa:didrocks/ubuntu-developer-tools-center
```

After years of seeing Ubuntu as a newbie-friendly distro, it seems strange to be talking of Utopic Unicorn as a distro for more technical users, but it seems to be the more experienced Linux users who will benefit most from this release. As well as *UDTC*, there are a number of rapidly developing new technologies that have been updated:

- Containers
- Juju



KDE users with a long memory will be wary of a new major version, but Plasma 5 on Kubuntu looks good, even if it's not the default version yet.



The Dash is Unity's most powerful feature. It's a central place that enables you to search a wide variety of sources. Here, it's searching both installed, and available applications.

■ Openstack

While many people often think of Ubuntu as a desktop distro, it's also one of the most popular Linuxes for running cloud services, and it's Ubuntu's tight integration with technologies like these that have earned it this spot. Of course, not many sysadmins are going to be rushing to switch their servers to a distro that'll only be supported for nine months. However, anyone looking to develop on these technologies may find that Utopic is the best distribution for them right now even if they release on a different system later on. This release of Ubuntu is also a great place for anyone looking to learn these tools (and learning these tools is a great idea if you plan on being in the sysadmin or cloud computing world in five years time).

Keep calm and carry on

Overall, Utopic isn't an exciting release for most desktop users. In fact, we'd go so far as to recommend most desktop users stick with Trusty Tahr (the most recent LTS version) for now. The exception to this is anyone interested in the developer tools, whether this is the *UDTC* or the cloud computing software. For these people, Ubuntu is becoming a very attractive option. That doesn't mean it's time to write Ubuntu off as a boring distro. There are big changes afoot. It's too soon to say whether they'll be for better or worse, but this will probably be the last boring Ubuntu release for quite some time. 

LINUX VOICE VERDICT

An unexciting release with a short support period... but be prepared for big changes to come.



FRITZ!Box 7490

Graham Morrison's distrust of his ISP's standard router has led him to a German alternative with a rather self-deprecating name.

DATA

Web
<http://en.avm.de>
Manufacturer
 AVM Computersysteme
Price
 £240

SPECIFICATIONS

VDSL or ADSL
 IP or analog phones
 4 x gigabit Ethernet
 WAN on LAN 1
 Wireless AC (1300 Mbit/s)
 Wireless N (450 Mbit/s)
 2 x USB
 Handles 6 DECT handsets
 ISDN S bus
 Average power: 9.3W

This is a beast of a router. It's also quite expensive and nowhere near as open as a device running the OpenWrt distro for embedded devices. But it's worth considering for a couple of reasons.

Most importantly, it can be used to replace the devices supplied by the UK's biggest fibre provider, BT Infinity. BT provides a proprietary and vendor specific unit it calls the 'HomeHub 5' to its customers – a wireless access point and VDSL modem combined. They're powerful, stable and more than adequate, especially in the fifth revision now being sent out. But BT also has a dubious record with its customer's privacy, including reports of 'remote diagnostic tests', its collusion with a commercial service that scanned your network packets to deliver tailored advertising, and rumours of BT allowing a government backdoor, NSA-style, into its hardware. The second and less tinfoil-hat reason is that the FRITZ!Box 7490 is much more powerful, giving you far greater control and feedback over your connection and replacing a multitude of devices within your home.

The most important feature for a device like this is the speed and stability of its internet connection. Every other feature hangs off its success, and we'd even forgo our privacy issues and stick with the HomeHub if it made the difference between watching Taylor Swift's videos in 1080p/60Hz, 24 hours a day, or ordinary standard definition at tea time. We've tested the unit with both a slower ADSL connection and a fibre-based VDSL connection, and we had good results over weeks of use. VDSL suffered from

occasional unrecoverable errors that the HomeHub didn't, but equally, it could go for weeks without a single problem. A firmware update that came late October (6.20) seemed to improve stability again, and added more feedback about the connection.

Such numbers, wow!

The feedback you get from the FRITZ!Box is fantastic. There are more geeky statistics on the quality of your connection than on any other router we've seen. An online monitor shows both a downstream and an upstream chart split by category of the data flowing into and out of your network. This is very useful if you want to immediately see how much bandwidth Netflix is consuming, as well as how much headroom you've got for playing *Borderlands 2* with your friends. There are charts showing the signal-to-noise ratio across the broadband spectrum, a page of statistics, and a page that allows you to adjust the signal-to-noise ratio, impulse noise protection and radio frequency interference to create a faster or more stable connection. These had no effect with VDSL, but they were able to make a troubled ADSL connection more stable for us.

We also like the simplicity of Access Profiles. These are a basic whitelist/blacklist system that can be grouped into profiles and applied to specific devices, or guest networks, on your network. You can filter on applications and by websites and block them completely or for a specific day and time. These lists provide a no-fuss way of locking down a child's device or cutting off their connections at a specific time. The firewall is equally well specified, allowing services and ports of your own creation through to specific devices, and there's a great overview of what devices are connected, what they're doing across the network, and at what speed they're connected. Other network features include access to the NAS and a portal that can make your router configurable from the internet. The NAS feels a little simplistic, and we don't think it's going to challenge QNAP and Synology devices, but it offers most features you'd expect, including access control and UPnP for media streaming. Just plug in a USB storage device and go.

Cable metropolis

The unit has four Gigabit Ethernet ports, and the 4th can be used to access the 'Guest' profile to limit access to any services running on your LAN. Wireless is via both a 2.4GHz frequency network and a 5GHz network. Without scientific testing, we'd say coverage wasn't as good as the HomeHub, but we had better throughput performance, especially after restricting

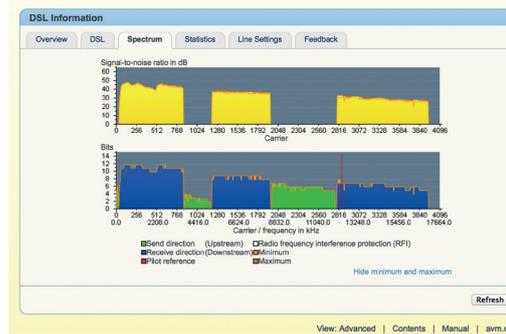


It looks like something out of *Elite Dangerous*, and the FRITZ!Box 7490 connected to our VDSL at the native speed for our line – 80 kbit/s download and 20 kbit/s upload.

Hack your FRITZ!box

There's a surprising amount you can do with the FRITZ!box if you're willing to hack around a little. Telnet can be enabled, for instance, by pressing '#' on a connected phone, and dialling 96*7*. You'll even see a message to tell you it's enabled (switch the 7 to an 8 to disable it). You can now connect to your router from the command line by typing **telnet ip_address**, and entering your web admin password. You'll now find yourself in a fairly recognisable Linux command line environment. You can **ls** or type **top** and take a look around the filesystem. **wget** is also available and there are ways of installing user-compiled packages like 'DropBear', but we wouldn't suggest doing this.

Instead, we'd recommend taking a look at the Freetz. This is a collection of open source modifications that can be made to your router's default firmware, turning it into a much more flexible and hackable device (and likely voiding the warranty at the same time). You need to create a virtual machine with a specific build environment and then pull the latest files from a GitHub account before building it all from a basic menu system. Nearly all the important and relevant information is in German, making the task that much harder if you don't speak the language. But if you want greater control over your



If the default firmware doesn't give you enough control, you can access the OS with Telnet or install an open source update called Freetz.

hardware, and the ability to make your own firmware changes, it's worth the effort. We found it particularly useful to add wider VPN functionality, including OpenVPN, as well as easily adding SSH and other applications.

5GHz to our faster devices. You can also turn the wireless off to a schedule and create a separate network for guests, which is a feature we really like.

There are several supporting apps you can install on your Android phone, and one is called 'WLAN'. This gives exceptional diagnostic information about the strength and frequencies of any networks overlapping yours, including the field strength over time and the ability to beep when it gets out of range. We found this an excellent tool for fine-tuning the network and making up for the weaker performance on the router.

As mentioned earlier, you can also pair the router with your DECT-compatible phones and connect your land line to your router to accept calls. You need to make sure you get the UK version for the UK cable, as it's a weird mix of broadband and phone that might not be easily found otherwise. You can enable both a fax and an answer machine on the FRITZ!box, and best of all, another app on your Android device will turn your mobile into yet another phone. When connected to the WLAN, the app will allow you to answer and make landline calls, listen to messages, make internal calls and see any missed messages, as can any other connected phone. This system works brilliantly, and we forgot we were answering a landline call from our mobile thanks to the app being so transparent. You can also add internet telephony



There are two USB 3 ports for printers and storage media, as well as analogue phone ports for telephones, answering machines and a fax.

to your configuration, turning any connected phone into an IP phone. We tested this with a free account from **www.voiptalk.org** and it also worked perfectly. You can block, divert and forward calls, depending on simple rules, and manage which network is used for dialling. If you want to trust the internet portal forwarding, you can even see your missed calls and messages from outside your network, making the box a good contender for a small business.

The best thing about the FRITZ!box is the huge amount of information and control it provides, and it's a refreshing change from the limited data

“The best thing about the FRITZ!box is the huge amount of information and control it provides.”

spat out by the routers shipped with most ISPs. For example, there's plenty of control over power usage, allowing you to reduce the speed of the network, or the wireless, or the DECT phones to save electricity. There are still trust issues with proprietary firmware, of course, and we missed specific 'Quality of Service' controls, but we feel a lot less paranoid while running the 7490 in place of our normal HomeHub 5.

This is an expensive router, but if you're looking to replace an old DECT phone system, a NAS server and media player, in a box that can handle FTTC and ADSL connection (including Annex M), or you run a small business, it pays for itself. 📺

LINUX VOICE VERDICT

Expensive, but more Linux-friendly than your ISP's default, and it can replace more than one device.



OpenBSD 5.6

Bang on schedule, another release of OpenBSD is here. Mike Saunders pits it against the current crop of Linux distros.

DATA

Web
www.openbsd.org
Developer
The OpenBSD Project
Licence
BSD, ISC, GPL and more

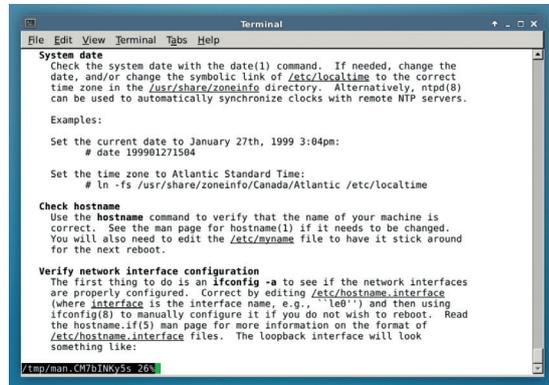
OpenBSD has a lot in common with Linux. It's Unix-like (actually a descendant of BSD Unix), it's open source, it's reliable, it's secure, and it runs most of the software you're familiar with: Gnome, KDE, Firefox, Apache, MySQL and so forth. But it's a much smaller project, with a comparatively tiny development team, so is it fair to compare it with Linux? We think so, given that it does a very similar job to Linux, and many OpenBSD advocates recommend making the switch. But we understand that it doesn't try to be an all-singing, all-dancing desktop OS like many of the big-name Linux distros.

OpenBSD 5.6 is available for several platforms, including x86, x86-64, PPC, Sparc and various ARM boards. An absolute minimum installation requires just 32MB of RAM and 200MB of hard drive space, so it's great for turning an old box into a router, firewall or small server – tasks at which OpenBSD excels.

Out with the old, in with the new

As with previous releases, the installer in 5.6 is plain text and to the point, and it's hard to get stuck because the documentation is so good. OpenBSD prides itself on having clear and correct documentation, and although its developers and users are often accused of chanting "RTFM" (read the fine manual) all too often, that's understandable. The installation guide is very thorough, and the "afterboot" manual page provides an excellent introduction to configuring your new OpenBSD setup. Very little is installed by default, but there's a large range of binary packages for desktop, office, server and development jobs just a few `pkg_add` commands away.

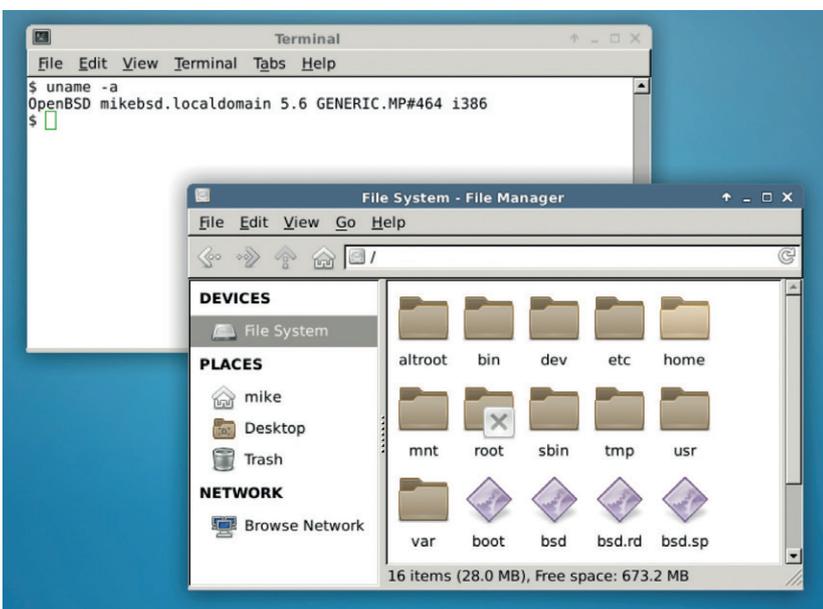
OpenBSD is pretty bare out of the box, but point `$PKG_PATH` at a mirror of packages and run `pkg_add xfce` to get this.



OpenBSD's manual pages are fantastic, and document virtually every aspect of the OS.

The biggest change in this release is the inclusion of LibreSSL, the OpenBSD project's fork of the hack-filled OpenSSL codebase. Some pundits criticised OpenBSD for forking and not simply submitting patches, but as we've seen with OpenSSH (another OpenBSD project that's now used everywhere), the developers know what they're doing. On laptops, the Intel and Radeon DRM drivers now have improved suspend and resume support, while *Sendmail* has been dropped from the base system in favour of *OpenSMTPD*. Other cleanups include the removal of *Kerberos* and the old *Apache*-based web server, with the latter being replaced by a small daemon based on *relayd*. Old-school FTP and tape installations have faced the chop as well in this release.

Now, OpenBSD prides itself on being extremely secure, beyond Linux and the other BSDs. We often see newly discovered exploits that wreak havoc on other OSes, but are more contained or simply don't work on OpenBSD. This is great, but keeping the system up to date is so much more work. OpenBSD doesn't have official binary updates – so when a bug or hole is discovered, you have to manually patch and recompile the kernel and/or userland yourself. There are some third-party providers of binary updates, but when you compare the work involved to `apt-get update && apt-get upgrade`, it's a bit off-putting. We understand that the OpenBSD team is small and very busy, so we won't knock them. It's just worth considering before making the switch. 



LINUX VOICE VERDICT

Plenty of old cruft removed, plus some minor updates. We still hope for binary updates one day though.



Google Cardboard

Ben Everard builds a virtual reality headset from an old pizza box. Forget jetpacks and hoverboards – this is the future.

The idea behind Google Cardboard is simple: take a mobile phone, place it close to your eyes and use one half of the screen to display an image to one eye, and another half of the screen to display an image to the other eye. This creates a stereoscopic 3D image that has half the horizontal resolution of the phone.

Google Cardboard does this with a piece of folded cardboard that fits onto your face and includes a pair of plastic lenses to help your eyes focus on the screen. There's also an NFC tag so your phone can tell it's in the device and a pair of magnets to provide input to the device.

You can buy kits with the cardboard pre-cut (from Amazon, **DX.com** or others), or you can just get the lenses and NFC tag (and magnets as well if your phone is compatible), then download the design files and make it from any pieces of cardboard you have lying about.

The biggest problem with Cardboard is how incompatible it is with many devices. More or less any phone should be able to display 3D images and video. The best source of these is probably YouTube (search for SBS 3D to find glass-compatible videos). Phones with a hardware gyroscope can also do head tracking, enabling you to look around the 3D world.

In fact, the gyroscope is probably the biggest problem with Cardboard at the moment. Not only do many phones (even some listed as 'partially-compatible' on Google's website) not have one, but those that do often have calibration problems that leave the 3D world slowly spinning (a problem known as gyro drift). It might be possible to correct (or at least minimise) this problem in software in the future, or this might prove too complex. For now, at least, this problem means that head tracking in Cardboard



is only really suitable for simple applications and not serious 3D work. This is a shame, and we hope that a solution is found.

What would MacGyver do?

The magnetic switch is probably the least reliable part of Cardboard. This should work with the Nexus 5, with other phones hit-and-miss at best. Without the magnetic switch, the method of getting input into the phone is to poke your finger through the nose hole and tap the screen. This is inelegant, but it works. Apparently, you can use a strip of copper tape to conduct a touch from outside the device to the screen, but we've been unable to test this.

Cardboard feels like (and it is) a hacky prototype stage. It's far from a consumer-ready device, and there's not a lot of software available for it yet. However, given the price it's selling at, we think that this is acceptable (gyro-drift aside). In fact, playing with Cardboard is ridiculously good fun partially because it feels so hacky. It literally is a virtual reality headset that you can build yourself, and that alone is enough to get us excited. There's just something about being able to use a device you've built to look around in a 3D world that restores our child-like wonder in computing far more than any closed off device you can pick up off a shop's shelf. We're giving Cardboard a maximum score because it's such a joy to use, not because it works perfectly. 

Anything that can output two images side by side can be used with Cardboard. This is the rollercoaster from <http://g.co/chromevr>

DATA

Web
<https://cardboard.withgoogle.com>

Developer
Google

Price
No official price. Various unofficial options from about £5.



The standard Cardboard (shown here) should fit most phones, but there's an unofficial large version that claims to fit phones up to 7 inches in size.

LINUX VOICE VERDICT

The most fun you can have with an old pizza box. Just don't expect everything to work perfectly.



Citizen Four FILM

Ben Everard didn't use Tor to book his cinema tickets and has now been flagged as a troublemaker.

In mid-2013, a trio of journalists boarded a plane to Hong Kong to meet a mysterious person who had recently delivered a small cache of classified documents from the NSA. The source promised to reveal more once they met. Those three people were Glenn Greenwald (a lawyer turned blogger and columnist), Ewan MacAskill (a reporter for the Guardian) and Laura Poitras (a documentary film maker). The person they met turned out to be Edward Snowden and the rest, as they say, is history.

Or is it? Many words have been spoken about Ed (as he likes to be known), but little has been said. Up until now, the three people who met him in Hong Kong have, for the most part, kept their reporting to the leaks, not the leaker. Although the mainstream media has dug through his past, he hasn't yet told his story, and while the history books are full of details of the data leaked, there are still some gaps in the public knowledge of the leak itself.

Citizen Four doesn't completely change that – it's ultimately a film about the actions of the American surveillance state, and Poitras's experience with that, rather than a biography of Edward Snowden – but it does offer a few glimpses round the side of the media firestorm. Poitras takes the viewer

directly into the hotel suite in Hong Kong for the meetings. We see Ed's reactions as the story blows up, but it's told subtly. Viewers have to watch closely to see how his body betrays his stoic words.

The footage of Snowden is interspersed with speakers on privacy and encryption including William Binney (the former NSA leaker), Jacob Appelbaum (Tor developer, journalist and privacy campaigner), and Ladar Levison (former owner of Lavabit). Together, these help give an idea of the impact that total surveillance can have on a society.

The human element

There's little about the details of the leak – it's been well covered elsewhere and film isn't the right medium to explore them. Instead, the film adds a human element to the saga. This has been much missed up until now (with, perhaps, the exception of the excellent book *The Snowden Files* by Luke Harding, but that book focused on the impact on the media establishments and journalists working at them, not the four people holed up in a crowded Hong Kong hotel room).

The geek viewer will appreciate that the technical scenes aren't faked (as so often seems to happen in films). When



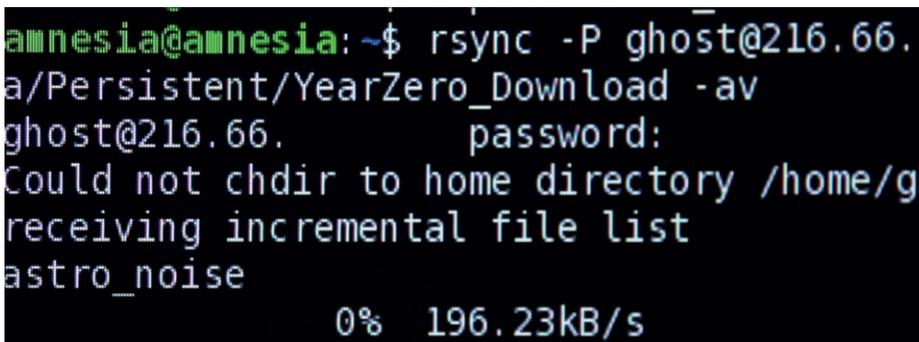
"My name is Edward Snowden. I go by Ed".

something is sent encrypted, we see **gpg** spit our seemingly random alpha-numeric characters (no, we don't know if these are actually the encrypted files). When it shows the command line, what else would you expect, but Tails':

```
amnesia@amnesia
```

These are of course small details, but the accuracy there gives us confidence in the accuracy of the film as a whole. Another nice touch was the inclusion of *Tor*, Tails and Debian Gnu/Linux in the film's credits. Eagle-eyed viewers will also learn that Ed's reading material of choice in Hong Kong was written by Cory Doctorow.

The film concludes with a confirmation of the much-rumoured new leaker from within the American surveillance establishment. Few details are given, but much is hinted at. Linux Voice attended the UK premier which included a Q&A session with Potrias where someone asked the director for any new details. Her refusal to answer gave us reason to suspect that we could be in for some more exciting (and worrying) news stories in the future.



We appreciated the accuracy of some of the technical details, like including a real command line rather than the fake ones you normally see whenever Hollywood touches on tech.

“Citizen Four is ultimately a film about the actions of the American surveillance state rather than a biography of Edward Snowden.”

LINUX VOICE VERDICT

Footage from inside the hotel room as Edward Snowden leaked the NSA's secrets. Need we say more?



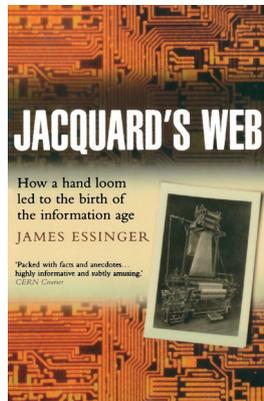
Jaquard's Web: How a hand loom led to the birth of the information age

19th century programmable clothing? Ben Everard is intrigued.

There was one interface that remained virtually unchanged from before Charles Babbage designed the Analytical Engine to well into the 1960's: punched cards.

These were first used by innovative weaver Joseph-Marie Jaquard to improve the speed at which silk could be woven into designs. These looms inspired Babbage in his ultimately futile quest to build a mechanical computer, they were instrumental in the pre-computer calculation machines, and they were considered essential components in almost all early computers.

Essinger follows this narrative and encourages us to follow the links from looms that weave silk to machines that weave information. Overall, it's an enjoyable read, and even people with quite a bit of knowledge of computing history will probably learn something from it.



Don't be fooled by the cover – surprisingly little of this book is actually about weaving clothes.

LINUX VOICE VERDICT

Author James Essinger
 Publisher OUP Oxford
 ISBN 978-0192805782
 Price £9

A slightly unusual angle that forces us to think about the data rather than the machine.

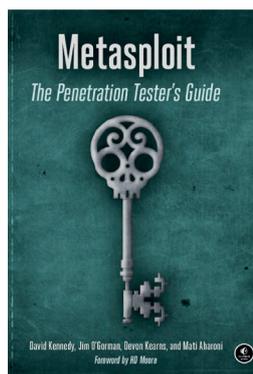


Metasploit: The Penetration Tester's Guide

Ben Everard learns how to exploit computers using PDF files.

Anyone interested in computer security will have come across *Metasploit*. It's a framework that can help with just about every aspect of penetration testing from information gathering to post exploit work. However, to do all this, it has become quite a complex beast. Learning to use *Metasploit* is more like learning to use a new programming language than learning to use most other applications.

Metasploit: The Penetration Tester's Guide is well titled: it's a guide to *Metasploit* for penetration testers. It isn't written for people with no pen testing experience, but it does start at quite a basic level and while it assumes the reader understands basic principals about how computers and networks work, it doesn't assume any prior experience with *Metasploit*. It's a very practical book, and a large proportion is taken up with code examples that are well annotated and explained.



HD Moore – the founder of the Metasploit project – endorses this book in the forward.

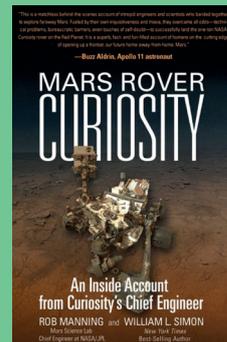
LINUX VOICE VERDICT

Author David Kennedy, Jim O'Gorman, Devon Kearns and Mati Aharoni
 Publisher No Starch Press
 ISBN 978-1593272883
 Price £32.50

A great introduction to the software, but you'll need some knowledge of penetration testing to get the most out of it.



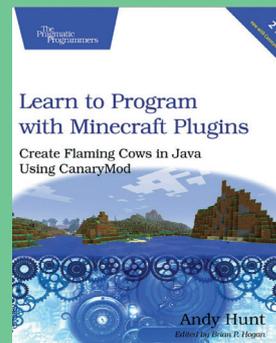
ALSO RELEASED...



Curiosity has spent over a Martian year (687 Earth days) on its new home .

Mars Rover Curiosity

If you've built Ben's ice cream carton robot from issue 4, and you're looking for a new challenge, this book is going to provide the perfect insight. Written by Curiosity's Chief Engineer and endorsed by Buzz Aldrin, it's got to be a must read.



The 2nd edition dumps Bukkit in favour of the CanaryMod library

Learn to program with Minecraft

Kids love *Minecraft*. It's amazing. This second edition of a proven child-friendly title promises that no prior coding experience is necessary, and that your firstborn will soon be creating flaming cows, flying creepers, teleportation, and interactivity.



SVG is a long way from the turtle graphics we cut our teeth on.

SVG Essentials

SVG is about more than being a scalable image format. It can be used for animations, charts, web design and includes transforms and gradient, all from its text-based XML. This is a book aimed at designers that doesn't ignore the complex stuff, making your output visually and functionally more powerful.

LINUX VOICE

LIGHTWEIGHT
DISTROS

GROUP TEST

Mayank Sharma is on the lookout for distros tailor made to infuse life into his ageing computers.

On Test

Linux Lite

URL www.linuxliteos.com

VERSION 2.0

DESKTOP Xfce

Does the second version of the distro does enough to justify its title?

WattOS

URL www.planetwatt.com

VERSION R8

DESKTOP LXDE, Mate, Openbox

Has switching the base distro from Ubuntu to Debian made any difference?

SparkyLinux

URL www.sparkylinux.org

VERSION 3.5

DESKTOP LXDE, Mate, Xfce and others

Multiple flavours with different desktops – is there one for you?

LXLE

URL www.lxle.net

VERSION 14.04

DESKTOP LXDE

What can this Ubuntu-based distro do that its parent cannot?

TinyCore

URL www.tinycorelinux.net

VERSION 5.4

DESKTOP FLWM

Will it live up to its name?

Slacko Puppy

URL www.puppylinux.org

VERSION 5.7

DESKTOP JWM

It's the leading distro for old computers, but can it win this challenge as well?

Lightweight distros

There has always been a demand for lightweight alternatives both for individual apps and for complete distributions. But the recent advent of feature-rich resource-hungry software has reinvigorated efforts to put those old, otherwise obsolete machines to good use.

For a long time the primary migrators to Linux were people who had fallen prey to the easily exploitable nature of proprietary operating systems. Of late though we're getting a whole new set of users who come along with their healthy and functional computers that just can't power the newer release of Windows.

Plenty of hardware is good enough to run the mainstream Linux distros without any issues. However, the modern Linux desktop is a fairly resource hungry beast as well, and your hardware might not have enough juice to power Unity or Gnome 3. In addition to users who have hardware that's been outdated fairly recently, there's another kind who are holding on to their workhorses from the last decade. They usually just use their computer to browse the web, do

some text editing, and watch some videos. These users don't need the latest multi-core machines loaded with several gigabytes of RAM or even a dedicated graphics card. However, chances are their hardware isn't supported by the latest kernel, which keeps dropping support for older hardware that is no longer in vogue, such as dial-up modems. Back in 2012, support for the i386 chip was dropped from the kernel and some distros, like CentOS, have gone one step ahead and dropped support for the 32-bit architecture entirely.

New life

In addition to pruning the resource requirements of mainstream software, a large number of open source developers are working to make obsolete hardware usable again. There are lightweight apps that consume a fraction of the resources of their full-featured alternatives and distros that perform blazingly well on low-spec machines. In this group test we look at some of the best distros that are designed from the grounds up to use the meagre hardware resources on dated computers.

DATING HARDWARE

Defining hardware as "older" is tricky. New software is always leveraging on the pace of hardware developments and rendering even relatively newer hardware obsolete. Examples of these relatively recent attic-ready hardware would be single-core or dual-core AMD Athlons and Intel Pentiums with about 1GB of RAM that they share with

onboard graphics. A couple of years ago, mainstream distros would perform adequately on these machines, but not anymore. Nowadays, you need special purpose distros to make good use of such hardware. We tested these distros on a 1.6GHz Intel Atom netbook with 1GB of RAM and on a dual-core 2.1GHz laptop with 2GB of RAM.

Lightweight alternatives

There are two things that are common to every lightweight distro – a lightweight desktop environment and/or lightweight apps. There are lightweight alternatives for every type of software. Office apps, such as the *AbiWord* word processors and the *Gnumeric* spreadsheet, are a popular replacement for mainstream full fledged suites like *LibreOffice* and *Calligra* and are still being

developed. Other featherweight contenders include the *Midori* web browser, *Xpdf* PDF viewer, *Fotocx* image editor, *CMPlayer* media player, and *App Grid* software centre, among others. Some distros, most notably *Puppy Linux*, also replace the stock apps with custom ones optimised for the minimal environment they are running under.

Similarly, there are also desktop environments that are lighter on resources

than *Unity*, *Gnome 3* and *KDE 4*. *LXDE* and *Xfce* are two of the popular options that are used by many distros. There's also *Mate*, which is a continuation of *Gnome 2*. Then there's *Enlightenment*, which includes some bling as well. Some distros, instead of using a full-fledged desktop environment, just use window managers such as *Openbox*. Other lighter but esoteric window manager are *IceWM*, *Fluxbox*, *FVWM*, and *JWM*.

WattOS R8

Desktop computing for cheap.

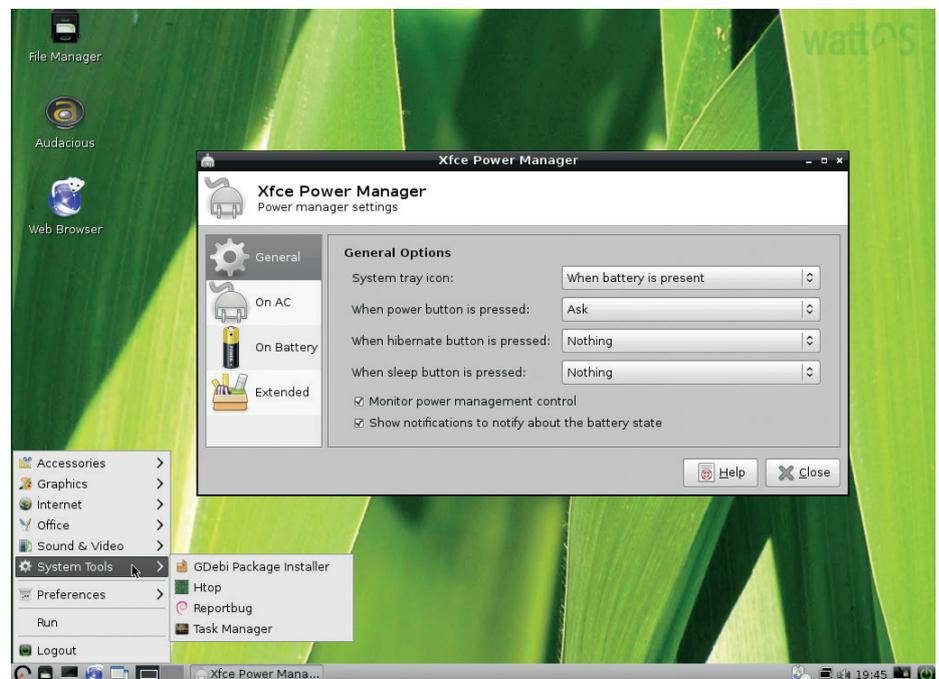
For a long time *WattOS* delivered the goodness of *Ubuntu* to low-powered machines. However, the distro has now switched to *Debian* and is based on the current stable *Wheezy* branch. It's available in three flavours: there's one with the *Mate* desktop and another with *LXDE*, which both weigh in at around 800MB. There's also an even lighter *Microwatt* edition.

Being based on *Debian*, installation duties are handled by *Debian's* graphical installer, which is pretty simple and straightforward. The *WattOS* developers have modified the installer with tips borrowed from the *LMDE* and *Point Linux* projects, but it still isn't as intuitive as *Ubuntu's Ubiquity* installer, with a manual partitioning step that might confuse new users. Once installed the distro occupies about 3GB of hard disk space.

Both the *LXDE* and the *Mate* flavours ship with the *Iceweasel* browser, which is *Debian's* trademark-stripped fork of the *Firefox* browser. Both flavours also include the *Shotwell* image organiser, *FileZilla* FTP, *Transmission* BitTorrent client, *Audacious* audio player and *Xfburn* media burner. The *LXDE* flavour also includes the *VLC* media player, which is oddly missing from the *Mate* version.

Missing in action

One app that was conspicuous by its absence was an office suite. All you have are basic text editors – *Mate* has *Pluma* and *LXDE* has *Leafpad*, and these are both quite stripped down. On the upside though you have the full *Debian* repository at your disposal that you can install additional apps from via the *Synaptic* package manager.



The distro also includes power management utilities like *PowerTOP* to optimise power consumption.

Both distros were quick off the blocks and we got to the desktops pretty quickly with *Mate* leading *LXDE* by a good 10 seconds. Once loaded though *LXDE* took up about 100MB less RAM than the fast-booting *Mate*. App launch times were fairly similar and both flavours performed as advertised. Even after several hours their memory footprints remained impressively low.

If you want something lighter still, there's the *Microwatt* edition. Instead of a full-fledged desktop manager, this flavour uses a customised *Openbox* window manager in place of the *PekWM* window manager in earlier versions. The bare-minimum desktop

sports the *conky* system monitor and a simple panel at the bottom of the screen that houses open windows.

The developers have done a wonderful job in picking up the right apps that don't gorge on resources and don't compromise usability as well. While the distro might be designed for older computers, we wouldn't shy away from recommending it as a lighter *Debian* alternative for regular use.

VERDICT

A wonderful distro that could put a spring in the step of any computer.

★★★★★

Linux Lite 2.0

Can lite be right?

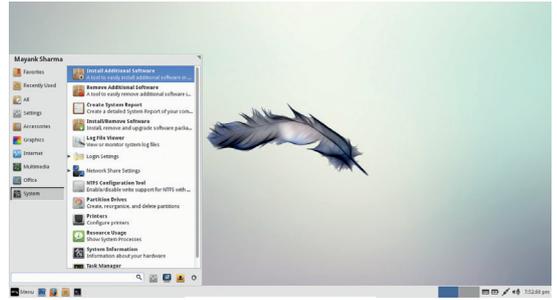
The Linux Lite distro is based on Xubuntu 14.04 LTS, which brings along two very important components – Xfce 4.11 and the excellent and easy-to-handle Ubiquity installer. One of the first things you'll notice after installation is the distro's beautiful login manager followed by the aesthetically pleasing all-white theme. The desktop is clean and uses the Xfce desktop's Whisker menu. The developers have tweaked the desktop, and the right-click menu includes options to create shortcuts, launch the task manager, and take screenshots.

At first glance, Linux Lite looks and feels like a regular heavy-duty distro. The list of pre-installed apps doesn't include any of the traditional lightweight apps and is instead brimming with the usual suspects such as *Gimp*, *Firefox*, *VLC* and *LibreOffice*. There's also the *Mumble* open-source VoIP software that's often compared with *TeamSpeak*

(it's also preconfigured here to connect to the **LinuxDistroCommunity.com's Mumble** server, so you can engage with other community users immediately.

Another unique aspect of the distro is its collection of well-designed scripts to add and remove programs. The Install Additional Software script gives a list of 25 apps, including *Google Chrome* and *Chromium* browsers, *Google Talk Browser Plugin* to video chat via Google Hangouts, *Dropbox*, *Netflix*, *Skype*, *Wine*, *PlayonLinux*, and more. There's also a corresponding Remove Software script that'll remove any software installed using the previous script.

There's also the *Synaptic* package manager and the distro is tuned into



Linux Lite is wonderfully stocked for everyday use, but this comes at the cost of relatively sluggish performance.

Ubuntu 14.04's official repository as well as several PPAs. This is also the first release which uses the Linux Lite's own repositories for managing its custom apps. One of these custom apps, the *Lite User Manager*, helps make managing accounts a little easier for new users to comprehend. New users will also appreciate the bundled support and help manual.

“One unique aspect of Linux Lite is its collection of scripts to add and remove programs.”

VERDICT

A good distro for someone who's new to Linux, but not necessarily lite.

★★★★★

LXLE 14.04

Luxuriously light.

There's a lot of similarity between Linux Lite and LXLE. Both are based on the latest Ubuntu 14.04 LTS release and neither include any of the usual lightweight apps, instead relying on their desktops. In that aspect, LXLE, which is based on Lubuntu and uses the LXDE desktop scores over Linux Lite by trimming almost 10 seconds of the latter's boot time. However, installing LXLE takes a lot longer than other distros. But that isn't a surprise considering it installs almost 2,000 packages.

The desktop is a pretty standard LXDE affair. It includes a panel at the bottom and a Unity-like launcher on the left that reveals itself when you mouse-over that part of the desktop. Some important information about the system is also displayed on the desktop via the *Conky* system monitor. However, the distro uses the malleable *ROXTerm*

as its terminal emulator rather than Lubuntu's *LXTerm*.

The distro is overflowing with apps. Besides the usual apps such as *Gimp*, *Shotwell*, *Claws* email, *FileZilla*, *Pidgin*, *Transmission*, *LibreOffice*, there are several unusual ones like *KeePassX*, *Marble*, *Anki* memory training, *Simple Image Reducer*, *BitTorrent Sync*, *Gitso VNC*, *Linphone*, *Homebank*, *Osmo*, *FB Reader* ebook reader, *BleachBit*, *Florence Virtual Keyboard*, and the *GSpeaker* frontend for *espeak*.

In stark contrast to many of its peers, the distro also includes several multimedia apps such as *Audacity*, *Arista* transcoder, the *OpenShot* video editor and *Minitube* for watching YouTube videos on the desktop. Another pleasant surprise is the inclusion of several games, from simple card games to the Steam client. For package management, the distro



The distro includes a whopping 100 wallpapers, and you can cycle through them with a button in the bottom panel.

includes both the *Lubuntu Software Center*, which is LXDE's answer to the *Ubuntu Software Center*, as well as *Synaptic*. Both are configured to fetch packages from the official Ubuntu repos and several other PPAs. LXLE also includes a graphical PPA manager using which you can add and manage additional PPAs.

VERDICT

Despite its long list of apps, the distro performs well on hardware with limited resources.

★★★★★

SparkyLinux 3.5

Give your computer a jump start.



SparkyLinux includes the *SparkyCenter* settings manager to house its custom apps.

Unlike other Debian-based distros in this group test, SparkyLinux is based on the distro's testing branch (called Jessie) and also borrows Debian's installer. SparkyLinux is available in several flavours, each based on a separate lightweight desktop environment. While the size of the ISO image and the number of installed packages vary from one flavour to another, installing them all took longer than the other distros in this group test.

Another common factor between all the SparkyLinux flavours is its collection of custom apps. There's the *SparkyAPTus* app, which is a simple front-end to the command line **apt-get** and **dpkg** tools. You can use this tool to edit the package repositories, install and remove apps and upgrade the whole system with a single click. *SparkyAPTus Extra* is a new app with which you can install popular apps such as *Dropbox*, *Skype*, *Steam*, *Tor Browser*, and more with a single-click.

The distro also includes custom apps to back up and restore app settings. All you have to do is select the app whose settings you wish to back up, and the app saves them in a compressed archive. You can then use the complementary *Restore* app to point to this archive to restore the settings. There's also an app to securely and permanently delete files, a *Wine* wrapper to install Windows **.exe** files, a manager for the *Conky* system monitor and tools to manage users and passwords.

Being based on Debian, the distro also includes Debian's forked trademark-free version of *Firefox* and *Thunderbird*, namely *IceWeasel* and *IceDove*. The browser is equipped with plugins to handle all kinds of content including Flash, QuickTime, DivX, RealPlayer and more.

Shiny desktops

The Enlightenment flavour uses the latest E19 desktop. However, the live environment performed terribly on the netbook and took aeons to install. Even afterwards moving the mouse wasn't smooth and animations were jerky; so, the Mate, LXDE, and Xfce flavours are the distro's leading contenders. You can use them in various languages and they all have a similar looking desktop with the *Conky* system monitor and the *Wbar* quick-launch dock at the bottom.

The distros are also loaded with apps including *LibreOffice*, *PlayonLinux*, *Gimp*, *Camorama* webcam, *Hotot* Twitter client, *gFTP*, *Pidgin*, *Gnome MPlayer*, *RecordMyDesktop* screencaster, *VLC* media and more. For system admins there's *GParted*, *Boot Repair*, a graphical front end to manage systemd and a graphical front end to install windows wireless drivers via *Ndiswrapper* and the *Gufw* firewall.

VERDICT

Performs admirably well on resource-deprived computers despite its gamut of apps.

★★★★★

Other options

The low-fat graphical options that aren't necessarily aimed at older hardware

There are several other distros that use the lightweight desktops that are often included on the distros we've tested here, not because of their size benefits, but for their similarity to the classic desktop metaphor, such as the Mate-powered Debian-based Point Linux.

Another lightweight distro is Elementary OS, which uses its own custom desktop environment along with a host of other custom tools, and is popular for its Mac OS X-like look and feel. However, the distro is designed primarily as a replacement for Windows and Mac, and wouldn't perform well on dated hardware.

Some distros – most notably Ubuntu – have official spins based on LXDE and Xfce, namely *Lubuntu* and *Xubuntu*, that are designed for hardware that doesn't have the power to run the main edition. There's also *Trisquel Mini* (a cut-down version of *Trisquel*, the distro made entirely with free software – even the non-free elements of the Linux kernel have been removed) for the free software purists, and *VectorLinux Light*, which uses *JWM* and *Fluxbox*. Another distro that uses *Fluxbox* is the Debian-based *AntiX*, which also relies heavily on custom tools.

The *SliTaz* distro also uses a bunch of custom tools but needs to be configured and set up before it can be used as a functional desktop. Finally, there's *Porteus*, which is primarily designed for installation on removable mediums like USB disks and CDs, but can also be installed on to a hard disk. The distro is small because of its modular nature and incredibly fast since it runs from the RAM, but it does involve a learning curve.



Check out *PepperMint Linux* if your bandwidth is good enough that you can live off web apps.

TinyCore 5.4 vs Puppy Linux 5.7

The most esoteric distros of the lot.

These two distros are arguably our first choice for powering computers that are light on resources. However, both distros are designed for advanced users, TinyCore more so than Puppy. Also, while both distros will revive literally any computer, their approach to addressing the problem is different.

TinyCore's sole developer, Robert Shingledecker was earlier involved with the once-popular but now-dormant Damn Small Linux project. The distro is available in three flavours. There's a miniscule 9MB core edition designed for servers, followed by the recommended TinyCore edition that weighs in at 15MB, and a 72MB CorePlus edition that has additional drivers for wireless cards, a remastering tool and internationalisation support.

True to its name, TinyCore bundles just a terminal, a text editor, and an app launcher on top of the light-weight FLWM window manager. It has a control panel to manage bootup services and configure the launcher. If you need anything else, you'll have to pull it in using the distro's package manager, including the installer if you want to install the distro full-time on to your hard disk.

Due to its minuscule size, TinyCore boots blisteringly quickly. But the distro's stellar performance comes at the price of usability. For starters, you'll be spending quite a lot of time with its package manager, which isn't the most

intuitive in the business. You'll also have to browse through its documentation to familiarise yourself with the distro's peculiarities, irrespective of your experience with Linux.

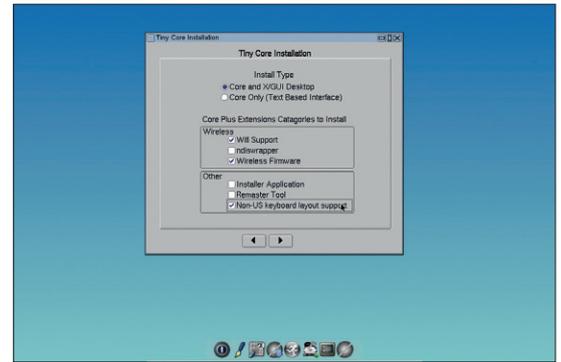
Puppy Linux

Puppy Linux's approach is in stark contrast to TinyCore. The project currently has three official versions. There's Wary Puppy, which is designed especially for older hardware, followed by Lucid Puppy, which is built from Ubuntu's binary packages; and Slacko Puppy, built from Slackware. Slacko is the recommended distro and the current 5.7 is built on of Slackware 14.

Instead of a full-fledged desktop manager, Slacko uses JWM, which is one of the lightest window managers. Installation is handled by Puppy's custom installer, which is well documented but has enough peculiarities to intimidate first-time users. For starters it lacks any automated partitioning tool and instead fires up *GParted* for the user to manually format the disk.

To its credit though, *GParted* is very verbose and each step has enough documentation for the user to carry out the task at hand. Also, once the installation has completed, users will have to manually invoke a tool to install the *Grub* bootloader.

There's no beating Puppy for out-of-the-box functionality and there's an app for virtually every task that you



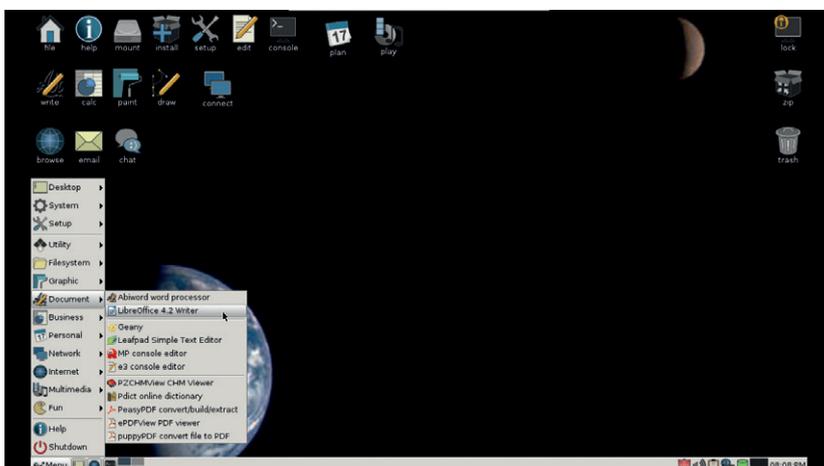
TinyCore includes a remaster tool with which you can create your own remix of the distro.

can perform with a desktop computer. However, most of these are Slacko's own custom apps instead of the popular ones that users are familiar with. Slacko has all kinds of multimedia apps include graphics viewers and creators and apps to playback, edit and even create multimedia. Furthermore there are apps to block website ads and for internet telephony, a podcast grabber, a secure downloader, a DVD burning app and lots more. The distro

“There’s no beating Puppy Linux Slacko for out-of-the box functionality.”

ships with several multimedia players, including *Gnome MPlayer*, to play all sorts of media formats. The included *Firefox* browser is equipped with all kinds of plugins and the distro also has a custom app to download and install the Flash plugin.

Puppy also scores for its ample documentation. Slacko bundles help documentation on several topics such as working with *Microsoft Office* files, how to add codecs, software and more and contains links to the documentation pages for most of the apps in its menus.



Puppy is very explicit in its instructions and verbose in its output.

VERDICT

TinyCore: Ideal for all kinds of hardware but setting it up is an involved process. ★★★★★

Slacko Puppy: One of the lightest around, but loses out for relying on its custom apps. ★★★★★

OUR VERDICT

Lightweight distros

If this were a group test of distros specifically for older hardware, then Slacko Puppy would have been top dog followed by TinyCore, which is lighter but has a more involved setup process. However, the real test of a lightweight distro is on a machine that's not necessarily old, just not adequately stacked. This is why we rated the distros based on their performance on the dual-core netbook, and the table below reflects numbers as measured on this computer.

WattOS Mate edition was quick off the blocks, but lacked any useful apps. The LXDE edition of the distro does include VLC but loses snappiness. We've rated it lower than the slow-starting LinuxLite because at least the latter welcomes the user to a much richer desktop experience. In fact, LinuxLite would be a good distro to introduce new users to Linux.

This group test would have easily been won by LinuxLite. It's one of the fastest distros and loads to a very usable desktop straight out of the box. But it loses out to SparkyLinux despite being faster because of its memory usage,

which is almost twice that of the winner. However, if you have the RAM to spare, then LXLE is our favourite LXDE-based distro around and it's even faster than SparkLinux's LXDE flavour.

Sparktastic

SparkyLinux is available in a number of editions with almost every lightweight desktop environment and window manager. But the performance varies greatly between its various editions. The Enlightenment edition is fastest but lacks any real apps, and also felt lethargic, much like the Xfce flavour. The mouse movement on both editions was jerky, and navigating through the menus wasn't smooth with the sub-menus revealing themselves only after a brief pause.

The Mate and LXDE editions were very usable and highly recommended. On these editions, you could further trim the boot times by tweaking some defaults, like disabling the *Wbar* panel at the bottom. Even without any tweaks, the Mate edition of the distro offers the right compromise between boot times, memory footprint and pre-installed apps.

1st Sparky Linux 3.5 Mate

Version 2.0

www.sparkylinux.org

The perfect balance of speed and functionality in this Mate powered distros that's chock-full of apps.

2nd LXLE 14.04

Version R8

www.lxle.net

A close second to Sparky Linux and the best lightweight distro based on the LXDE desktop.

3rd Linux Lite 2.0

Version 3.5

www.linuxliteos.com

The distro works well to free up resources on newer machines but doesn't feel comfortable on older ones.

4th WattOS R8

Version 14.04

www.planetwatt.com

The Mate edition of this distro boots faster than all others but lacks any meaningful apps.

5th Slacko Puppy 5.7

Version 5.4

www.puppylinux.org

Our favourite distro for revising older computers, but it loses out in this challenge for its reliance on its custom apps.

6th TinyCore 5.4

Version 5.7

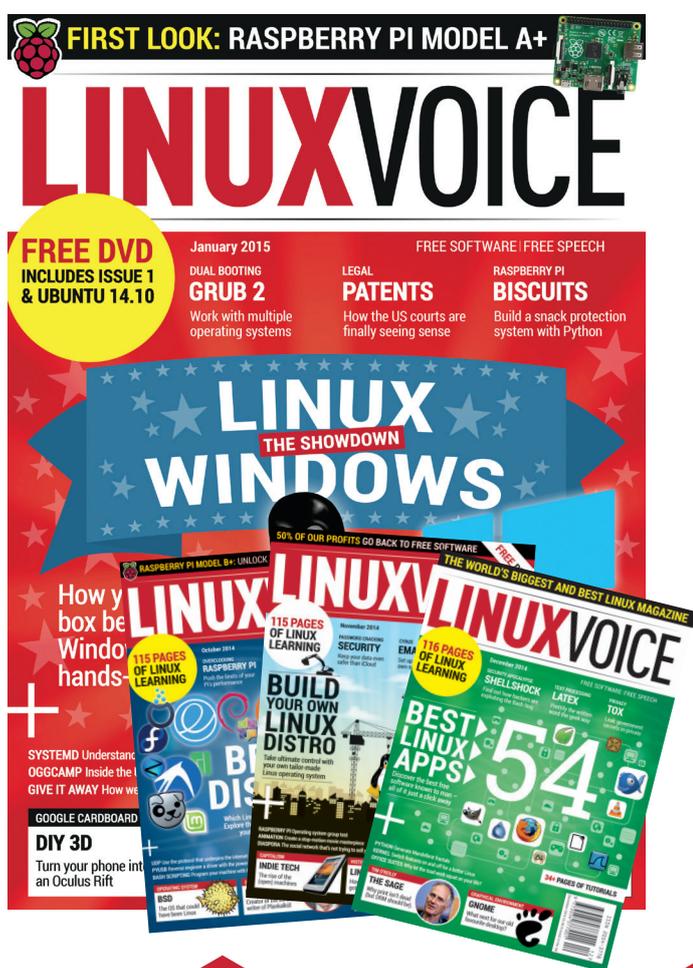
www.tinycorelinux.net

The lightest Linux distribution of them all, it loses out for its involved setup process.

	Boot time	Used memory	Total Packages	Desktop Environment	Package Management	Office Suite	Multimedia app
Linux Lite 2.0	41.2s	501 MB	1389	Xfce 4.11	Synaptic	LibreOffice	VLC
WattOS R8 Mate	14.6s	332 MB	1125	Mate 1.8	Synaptic	None	None
WattOS R8 LXDE	23.4s	231 MB	1118	LXDE 0.5.5	Synaptic	None	VLC
LXLE 14.04	34.5s	775 MB	1931	LXDE 0.5.6	Synaptic	LibreOffice	Several
Tiny Core 5.4	7.5s	46 MB	—	FLWM	Custom	None	None
Slacko Puppy 5.7	17.2s	112 MB	333	JWM	Custom	AbiWord, Gnumeric	Gnome Mplayer
Sparky Linux 3.5 e19	31.6s	285 MB	1267	Enlightenment e19	Synaptic	None	None
Sparky Linux 3.5 Mate	37.2s	357 MB	1894	Mate 1.8	Synaptic	LibreOffice	Several
Sparky Linux 3.5 LXDE	41.3s	342 MB	1994	LXDE 0.5.5	Synaptic	LibreOffice	Several
Sparky Linux 3.5 Xfce	38.2s	418 MB	2019	Xfce 4.10	Synaptic	LibreOffice	Several

SUBSCRIBE

shop.linuxvoice.com



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

LV Gives 50% of its profits back to Free Software

LV Licenses its content CC-BY-SA within 9 months

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

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

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

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

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

NEXT MONTH IN LINUX VOICE

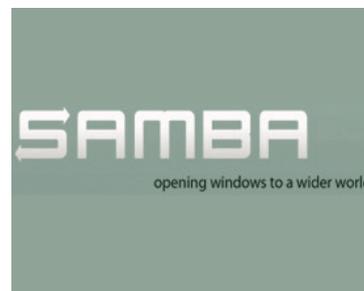
**ON SALE
THURSDAY
18 DECEMBER**



FREEDOM ISN'T FREE

We take it for granted that Free Software is important – well, we shouldn't. This is our manifesto. This is what we believe.

EVEN MORE AWESOME!



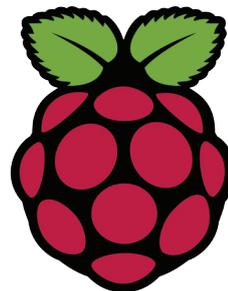
Samba 4

Now the power of Microsoft's Active Directory can be yours to command, as Linux admins, finally, can use it at work with Windows admin accounts.



Scribus

This free software tool is a superb design application, but how well can it integrate with a proprietary software workflow?



Picademy

Our mole at the Raspberry Pi Foundation has revealed something amazing: they're teaching IT teachers how to teach IT! Imagine that!

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 June 2015 when all content (including images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2014
ISSN 2054-3778

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



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

CORE TECHNOLOGY

Prise the back off Linux and find out what really makes it tick.

Access control

Get to grips with file access controls in Linux as we take “rwx” to the max.

Check into a hotel, and the chances are you'll get a little plastic card. Take the lift to your room, slot the card in the door, and either it will let you in or you will have to retrace your steps to the check-in desk to get a new card. That's access control. Within an operating system, it can be applied at various points. Consider the common scenario of browsing to a website that's delivered by a LAMP stack. Access controls are applied at several places in the data path:

- 1 Packet filtering rules in the Linux kernel can block incoming traffic except to a few favoured ports, or from a favoured IP address range perhaps.
- 2 The *Apache* web server can apply access controls based on directives in its config file. For example, “allow from” and “deny from” rules control access based on the client machine identity, and it also supports user logins to restrict access to parts of the site to specific authenticated users. Many servers such as Samba and Postfix can impose similar rules.
- 3 The PHP code on a web page queries a back-end database, probably *MySQL*. Here again, there's a need to authenticate, and there's a fine-grained set of privileges defined in *MySQL*'s grant tables.
- 4 Underneath it all, we have the Linux filesystem. Again, access control decisions are made based on the file's ownership and permissions, and on the identity of the process attempting the access.

Indeed, there are so many points along the way at which something might say “no” that it's a miracle it ever works at all. And don't even get me started on the mandatory access control mechanisms like SELinux and App Armor.

In my list, items 2 and 3 are implemented by rules in userspace (ie within the code of the server itself), whereas items 1 and 4 rely on decision-making code in the kernel.

It's the bottom layer (item 4 – file access control) that I want to focus on here. The detail of how this works is a well-travelled road and probably familiar to you. But I'll try to turn over a few stones along the way that you haven't looked underneath before.

Burn after reading

A good place to start is by examining a couple of lines from `ls -l`:

```
# ls -ld /home/chris /etc/shadow
-rw-r----- 1 root shadow 1284 Sep 29 13:51 /etc/shadow
drwxr-xr-x 93 chris chris 4096 Sep 29 12:08 /home/chris
```

The first field of the line includes the classic nine permission bits, the third and fourth fields show the owner and group of the file. The table ‘rwx explained’ shows the

meaning of these three permissions as applied to files and directories. The meaning of the execute bit, in particular, is totally different for files and directories. You should also notice that you need to have both read and execute permission to have any sort of useful access to a directory, and they invariably go together; you are very unlikely to come across a directory that has one but not the other.

This is pretty basic stuff. “Sure”, I hear you say. “I knew all that”. But do you really understand how it works? Consider this:

```
$ ls -l test
```

```
-r--rw-rw- 1 chris chris 0 Sep 19 16:10 test
```

My question is, can I (chris) as the owner of the file, write to it? Before you read on, have a guess!

Then try this:

```
$ echo hello > test
```

```
bash: test: Permission denied
```

...which will convince you that you can't. It is tempting to think that because members

Create a shared directory

Our mission here is to create a directory that can be used as a shared read/write resource by two users: andrew and graham. We will accomplish this by making them both members of the group “editors”, and the directory will be `/home/editors`.

First, create the group:

```
# groupadd editors
```

Now create the directory and set its group. Then set its `setgid` bit so that files created here will have the group “editors”:

```
# mkdir /home/editors
```

```
# chgrp editors /home/editors
```

```
# chmod g+ws,o-rwx /home/editors
```

It should now look like this:

```
# ls -ld /home/editors
```

```
drwxrws--- 2 root editors 4096 Sep 29 20:38 /home/editors
```

Now make sure that andrew and graham are members of the group (we assume they already have accounts):

```
# usermod -a -G editors andrew
```

```
# usermod -a -G editors graham
```

Finally, establish a `umask` of 007 for andrew and graham by adding a line:

```
umask 007
```

in the `~/.profile` file of the two users. This will ensure that files they create are accessible by the group but not by others. Allowing the files they create to be group writable is OK because outside of `/home/editors` any files they will create will have their regular primary group (andrew or graham) and only the accounts andrew and graham will have membership of those groups. For more on `umask`, see Command Of The Month.

of the file's group (and for that matter everyone else) have permission to write to the file, then you should too. But it doesn't work that way. If I am the owner of the file, I see the first set of permissions. End of story. The tests are not applied in a hierarchical way. In practice this situation almost never arises. It is very rare (and not particularly useful) for the permissions to become more liberal as you move from left to right (owner/group/other). It's usually the other way round.

The other interesting question is "Can the owner (chris) delete the file?" Have another guess, then try this:

```
$ rm temp1
rm: remove write-protected regular empty file
'temp1'? yes
$
```

You will get a warning from **rm** about deleting a file on which you don't have write permission, but if you answer 'yes' it will do it. The point is this: whether you can delete a file does not depend on the permissions on the file itself; it depends on the permissions on the directory you're trying to delete it from (you need write permission). This is logical if you think about it – we're not actually trying to write to the file, we're just trying to write to the directory (to remove the link). But in the training classes I run I usually see a few raised eyebrows and wrinkled noses when I explain this. The fact is, there is no "you can delete me" file permission in Linux.

The warning you receive from **rm** in this case is not characteristic Linux behaviour. The usual philosophy of these commands is "if you asked to do it, and if you have permission to do it, I'll do it". For example, if you have files **foo** and **bar**, neither of these commands will raise a query:

```
$ cp foo bar
$ mv foo bar
```

even though both of them will result in the loss of the original file **bar**.

The nine "rwx" mode bits on a file are well known, but there are actually three more,

Fool your boss!

Do you have to contend with an insufferable Linux know-it-all at work? Would you like to outsmart them? Then try this:

```
$ touch temp
$ chmod 7000 temp
$ ls -l temp
---S--S--T 1 chris chris 491 Jun 30 15:49 temp
```

Ask them to do an **ls -l** on the file then explain the permissions. Of course, there's a small risk that they'll actually know, in which case they'll become even more insufferable!

The rwx permissions: files vs directories

Permission	On a file	On a directory
r (read)	Read the file. (Make a copy of it, display it, compile it.)	List the names of the files in a directory.
w (write)	Write to the file. (Edit or append to it or copy over it.)	Create file (links) in the directory, or remove them.
x (execute)	Run the file as a command. (Applicable to compiled programs or to scripts.)	Access the files in the directory, use the directory in a pathname.

known as the "set user ID bit", the "set group ID bit", and the "sticky bit" which are less well understood. The "set user ID" bit (**setuid** to its friends) is probably the most interesting, and by way of introducing it, consider these three facts:

- 1 Passwords in Linux are stored in a file called **/etc/shadow**. (Actually, it's the password hashes that get stored.)
- 2 The shadow file is heavily protected – ordinary users can neither read or write to it.
- 3 Users can change their passwords without administrative help.

So how does this work, exactly? The following commands give us a clue:

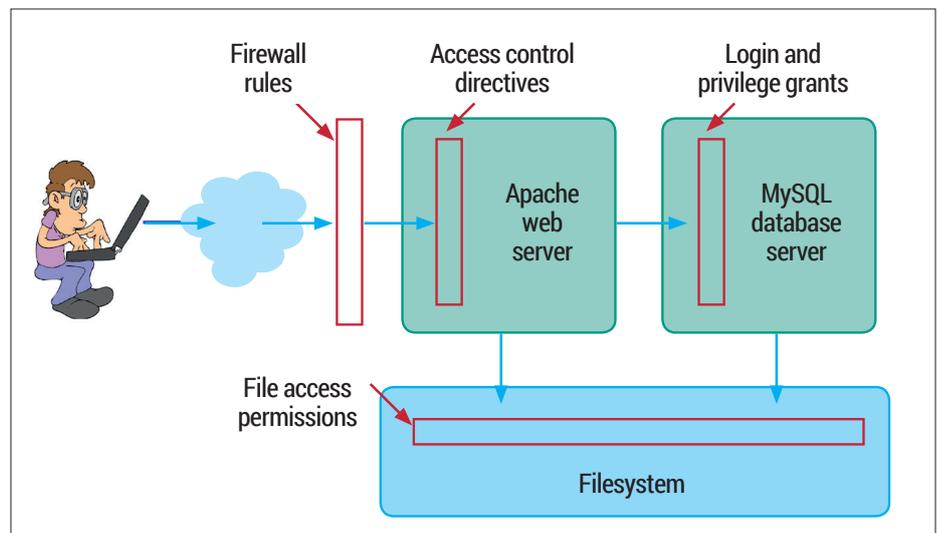
```
$ which passwd
/usr/bin/passwd
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 47032 Feb 17 2014 /usr/bin/passwd
```

The thing to notice is the **s** in the fourth character position, where the **x** would normally be. This is the famous **setuid** bit. Let me explain.

If a file is executable, and it has the **setuid** bit set, then while it is running it runs with

the effective privileges of its owner – in this case root. The program is said to run "setuid to root". As root, of course, it can do anything, including writing to the shadow file. Once the program terminates, you're back to your normal identity at the shell prompt. This mechanism lies at the heart of all "privilege escalation" events in Linux. It's how commands like **su** and **sudo** work, for example. It's also used by programs like **ping**, not because they need to elevate their access into the file system, but because they need to reach right down to the IP layer and construct unusual network packets, which only root processes can do.

The **setuid** bit was invented by the late, great Dennis Ritchie while he was working for Bell Telephone Laboratories. They obtained a patent (US patent 4135240 – look it up if you're interested) though they never charged licensing fees and later placed the invention into the public domain. Interestingly, the operation of **setuid** was explained on the patent by a diagram of logic gates and signals, it being unclear at the time whether a software algorithm could actually be patented.



Linux applies access control at several points within a LAMP (Linux, Apache, MariaDB/MySQL, PHP/Perl/Python) stack, some in the kernel and some in userspace.

Finding out who you are

We can demonstrate the effect of the **setuid** bit with a trivial C program using the **getuid()** and **geteuid()** system calls, which report on the real and effective UID of the process respectively. Here's the code:

```
#include <stdio.h>
main()
{
    printf("Real ID = %d\n", getuid());
    printf("Effective ID = %d\n", geteuid());
}
```

I won't insult your intelligence by dissecting this! Now compile and run it:

```
gcc uiddemo.c -o uiddemo
$ ./uiddemo
Real ID = 1000
Effective ID = 1000
```

No surprises here. 1000 is the UID of my regular account. Now try this:

```
$ sudo chown root uiddemo
$ sudo chmod u+s uiddemo
$ ls -l uiddemo
-rwsrwxr-x 1 root chris 8620 Sep 28 18:52 uiddemo
$ ./uiddemo
Real ID = 1000
Effective ID = 0
```

Now that the program is running **setuid** to root, it reports an effective UID of zero, and at this point, it is all-powerful.

The **setuid** bit is not honoured on shell scripts. To demonstrate this, begin by resetting the ownership and mode of **uiddemo** back to what it was:

```
$ sudo chown chris uiddemo
```

```
$ sudo u-s uiddemo
```

Now wrap the program in a tiny shell script called **uiddemo-bash** like this:

```
#!/bin/bash
./uiddemo
and make the shell script itself run setuid to root:
$ sudo chown root uiddemo-bash
$ sudo chmod u+s uiddemo-bash
$ ls -l uiddemo-bash
-rwsrwxr-x 1 root chris 24 Sep 28 18:57 uiddemo-bash
$ ./uiddemo-bash
Real ID = 1000
Effective ID = 1000
```

The output shows clearly that the script is owned by root and has the **setuid** bit on; nonetheless, no privilege change has occurred.

Programs that run **setuid** to root need to be coded with great care to avoid design flaws, buffer overflows, or other vulnerabilities which would put them at risk of being subverted.

You can find all the **setuid** programs on your system with the command:

```
$ sudo find / -user root -perm +4000 -ls
```

Hopefully, you shouldn't find too many. And if you claim to be a security-conscious system administrator, you really ought to be able to say what all of them are for.

The "set group id" (**setgid**) bit is similar. Whilst the program is running, it runs with the effective group of its owner. Again, **find** will locate these for you:

```
$ sudo find / -perm +2000 -ls
```

You'll probably see quite a few administrative group ownerships showing up in this list. Running **setuid** is potentially more dangerous than running **setgid**, and running **setuid** to root is most dangerous of all. But using **setgid** effectively takes more thought about setting file groups and permissions. And sometimes **setuid**-to-root is necessary to get the job done.

The **setgid** bit has a totally different meaning when applied to a directory. Normally, when a file is created, its group is taken from the primary group of its creator. However, if the file is being created in a directory that has the **setgid** bit set, the group will be inherited from that of the directory. (This obscure feature makes a wonderful Linux interview question, by the way!) See the box Create A Shared Directory for an example of how to exploit this feature.

The sticky bit

In the very early days of Unix, the "sticky bit" would be set on a frequently used executable to encourage the operating

system to keep it loaded in memory after use, thus speeding things up the next time it was run. These days, a program's code is paged in as part of the virtual memory management, and the sticky bit's original role has long been redundant. It has, however, found an important retirement job when applied to directories.

As we have seen, you can delete any file if you have write permission on the directory you're deleting it from. This doesn't work too well for world-writable directories (**/tmp** is the obvious example; you may find a few others) because you would be able to delete other peoples' files. The sticky bit changes the rules so that you can only delete files that you own.

The way that **ls -l** shows you these three extra bits (putting an **s** or **t** over the top of the 'x' bits) is confusing to say the least. It would have been better, perhaps, for **ls** to squander three more character positions on the line to represent them explicitly, but I guess there are too many scripts out there that make assumptions about the output

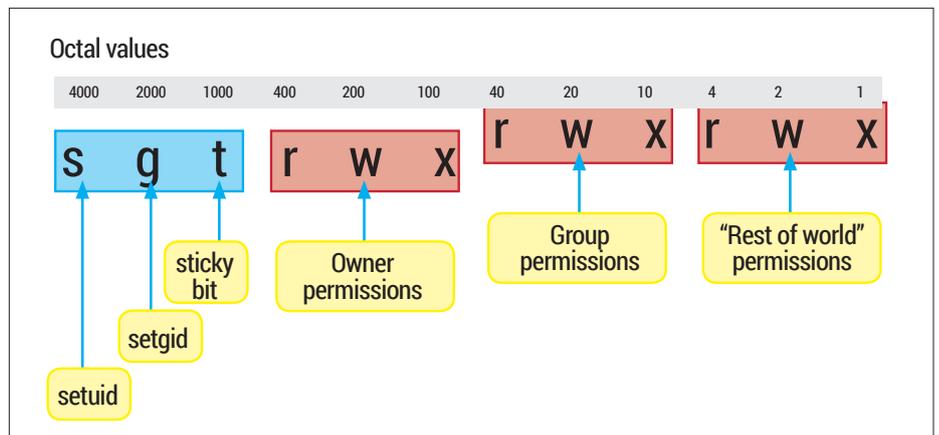
format from **ls -l** to consider changing it now. The assumption is that the 'x' bit that's underneath is set (for example it doesn't make sense to set the **setuid** bit on a file if it's not executable). If the underlying 'x' isn't set, you'll see an upper-case S or T instead (see the box "Fool your boss!") but you need to be pretty eagle-eyed to spot this.

The scoop on groups

Every user has a primary group. The primary group assigned to them when they log in is taken from the fourth field of **/etc/passwd**.

On many systems, adding a user account will, by default, add a new group with the same name, so that each user has a personal primary group that only they belong to.

However, it's easy to create new groups and assign a user any primary group of your choice when you create their account. A user can also have a number of secondary group memberships. These are defined in **/etc/group**. For example, if you see this line in **/etc/group**:



Every file and directory carries these 12 access control bits; each can be identified by its octal value.

```
editors:x:1002:andrew,graham
```

then andrew and graham have “editors” as a secondary group.

You can create a new group like this:

```
$ sudo addgroup editors
```

```
Adding group `editors' (GID 1002) ...
```

```
Done.
```

And you can give a user secondary group membership like this:

```
$ sudo usermod -a -G editors andrew
```

When a user creates a file, the file’s group is normally set to the user’s primary group (for an exception to this see the box Create A Shared Directory). When a user accesses a file that he’s not the owner of, if the file’s group matches any of the user’s groups, (primary or secondary) then he’ll see the group permissions on the file. In effect, you are “in” all your groups simultaneously.

Move users around

So for example, user andrew with primary group “andrew” and secondary group “editors” will be able to read the file `/tmp/target` with ownership and permissions like this:

```
$ ls -l /tmp/target
```

```
-rw-r----- 1 root editors 24 Sep 29 13:47 /tmp/target
```

It’s also possible for a user to switch his primary group to be any of his secondary

“It’s possible for a user to switch his primary group to be any of his secondary groups.”

groups, though this feature is rarely used in my experience. So, for example, andrew can switch his primary group to be “editors”, like this:

```
$ newgrp editors
```

The three “extra” mode bits -- less well known but very important

Permission	On a file	On a directory
s (setuid)	Run an executable file with the effective user privilege of its owner	Unused
g (getuid)	Run an executable file with the effective user privilege of its group	Files created in this directory will inherit their group from the group of the directory
t (sticky bit)	Unused	Only allow users to delete files from this directory if they own the file

This command starts a new shell with the new primary group identity. (And by the way, `newgrp` runs `setuid`-to-root to allow it to do this.) The following sequence of commands shows `newgrp` in action. Assume andrew has just logged in, and pay attention to the group of the files `f1` and `f2`:

```
$ touch /tmp/f1; ls -l /tmp/f1
```

```
-rw-rw-r-- 1 andrew andrew 0 Sep 29 14:33 /tmp/f1
```

```
$ newgrp editors
```

```
$ touch /tmp/f2; ls -l /tmp/f2
```

```
-rw-rw-r-- 1 andrew editors 0 Sep 29 14:33 /tmp/f2
```

You can set a password on a group, though hardly anyone uses this feature. If you do, a user can make any group his

```
# gpasswd hackers
```

```
Changing the password for group hackers
```

```
New Password:
```

```
Re-enter new password:
```

The password hashes are stored in `/etc/gshadow`, analogous to `/etc/shadow` for user passwords. Now, if andrew tries to make hackers his primary group, although he’s not a member, he’ll succeed if he knows the password:

```
$ newgrp hackers
```

```
Password:
```

```
$ id
```

```
uid=1001(andrew) gid=1004(hackers) groups=1001(andrew),1002(editors),1004(hackers)
```

Be aware, though, that telling someone the password for a group is not the same as making them a member of the group.

Although file permissions are not the only place that Linux applies access controls, they are absolutely essential to system security. In my experience a high proportion of system problems or security holes come down to misunderstanding or misapplying file permissions. Knowing how they really work is important. 

primary group as long as he knows the password. Let’s try. First, we’ll add a new group called “hackers”:

```
# groupadd hackers
```

Now we’ll set a password on it:

Command of the month: **umask**

Strictly speaking, `umask` isn’t a command, it’s a shell built-in, but I don’t mind thinking of it as a command if you don’t. The `umask` setting (usually established in a shell startup file such as `~/.profile`) is used to limit the permissions that will be assigned to any file you create. It’s a bit confusing because it sort of works ‘upside down’ – the permission bits that are set in `umask` will be withheld from any files you create.

Maybe a couple of examples will help. First we’ll set our `umask` to zero:

```
$ umask 000
```

Now if we create a directory and check its permissions we see something like this:

```
$ mkdir test1
```

```
$ ls -ld test1
```

```
drwxrwxrwx 2 andrew andrew 4096 Sep 29 21:20 test1
```

Notice the very liberal permissions on the directory. Now let’s tighten things up by setting a `umask` and repeating the experiment:

```
$ umask 027
```

```
$ mkdir test2
```

```
$ ls -ld test2
```

```
drwxr-x--- 2 andrew andrew 4096 Sep 29 21:20 test2
```

We see that the bits that are set in the `umask` (we might represent it as `---w-rwx`) are NOT set in the file’s permissions. To be exact, to compute the permissions that will be assigned to a file when it is first created, take the one’s complement of `umask` and perform a bit-wise logical **and** operation with the permissions requested by the program that created the file. Also, you should understand that `umask` only takes effect at the instant a file or directory is created. It doesn’t apply retrospectively to existing files.

FOSSpicks

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



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

Partition manager

GParted 0.20

Version numbers are funny things. Some developers would argue that they don't mean much, but end users often infer a lot from them. We know of quite a few FOSS programs that have surprisingly low version numbers, given their quality and feature set, such as *Inkscape*. It has been around for years, has loads of features, is used by professionals for real work, and yet it's only at 0.48. The same could be said of *GParted*: it's a hugely useful and mature tool, yet its 0.20 version number suggests it's barely at alpha stage.

That might scare some users away, especially given its job of performing (potentially risky) filesystem operations. In our experience over the years, though, *GParted* is a tool you can rely on. It's a GTK-based program for managing partitions on your hard drive, and it's available in two flavours: as a standalone app, or as a live distro.

The latter is what we're looking at here, as it's very useful to have around on a spare CD-R or USB key for emergencies.

The ISO weighs in at a touch under 200MB, and when booted, it asks for your keyboard layout and language. Then it drops you into a rather ugly Fluxbox desktop, with some dreadful desktop icons. Sure, glitz and fancy effects aren't important here, but a bit of work on the presentation would be welcome, especially when you're using it to fix someone else's system.

Anyway, *GParted* itself pops up, showing the layout of your hard drive. (If you have multiple drives, you can select the one you want to edit via the drop-down menu on the top-right of the window.) Click on a

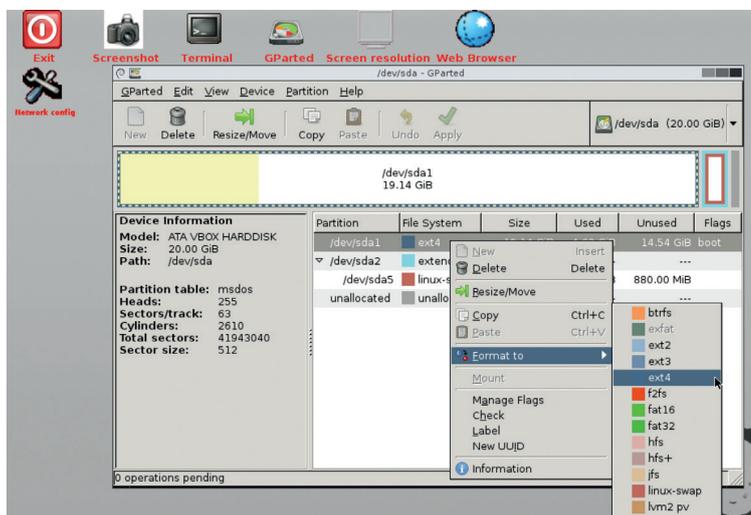
"GParted is very useful to have around on a spare CD-R or USB key for emergencies."



This panel shows the range of filesystems that are supported, and operations available for them.

partition in the list to select it, and then use the toolbar at the top (or the right-click menu) to perform an operation on it. Note that *GParted* doesn't perform its operations straight away – instead, it batches them up until you click the Apply button at the top.

GParted is tremendously versatile, supporting over 20 filesystem formats, although not every operation is available for each format. But for the common formats (ext*, btrfs, NTFS, FAT32, HFS+) you can create, copy, resize and relabel partitions. In many cases you can also perform checks on partitions, and attempt to recover deleted ones. It's a great toolbox for working with hard drives, and an essential part of a sysadmin's armoury.



The GParted Live distro isn't a feast for the eyes and looks very late 90s, but it gets the job done.

PROJECT WEBSITE
www.gparted.org

Operating system

Minix

Although it's not very well known, Minix played a major role in the early days of Linux. It was created by computer science professor Andrew Tanenbaum as a learning tool, and Linus Torvalds used it to build the very early releases of his kernel. The two hackers got into a fascinating online debate about kernel design – a debate that has become so famous, it even has its own Wikipedia page: <http://tinyurl.com/b2us8t>. Tanenbaum argued that microkernels are the future, and Linux was already obsolete before it had even taken off. Torvalds disagreed, of course.

But what is a microkernel? Essentially, it's a very small kernel that does a handful of vital jobs: mapping memory, controlling processes, and enabling processes to communicate with one another. Everything else, including hardware drivers, networking protocols and so forth, are run in "userspace", where they can't interfere with the inner workings of the kernel. Contrast this to Linux, where a good chunk of the OS's functionality is provided directly inside the kernel.

So a microkernel should, in theory, be more reliable, as parts can be swapped out more easily in the event of a bug wreaking havoc. But when you have more things happening in userspace, and major

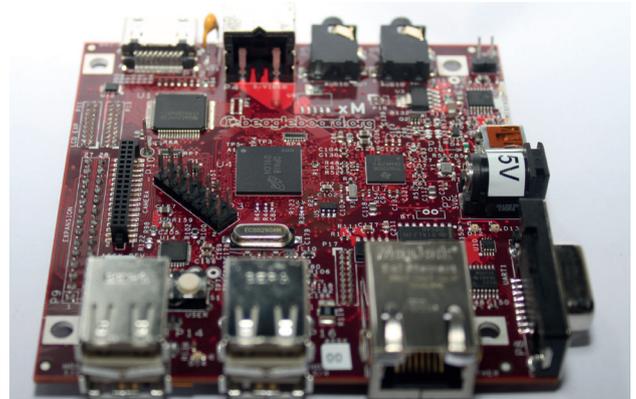
parts of the system exchanging messages all the time, performance can be impacted significantly. So the debates continue today, but Minix 3 is a good example of a genuinely useful all-round OS built on a microkernel design.

A bit of this, a bit of that

Minix 3 is now regarded as an alternative to Linux and the BSDs for embedded devices and low-spec hardware and it runs many FOSS apps you're familiar with. To install it, you'll need at least 64MB of RAM and an i586 or newer processor along with 675MB of drive space.

Minix 3.3 is provided as a 288MB compressed ISO image, which extracts to 578MB, so it can be burned to a regular CD-R. After booting, you'll be prompted to log in as root (without a password), and then run **setup** to begin installation. This is all plain text, and there are no hand-holding wizards, but if you have some Unix experience you won't find it too daunting.

By and large, Minix has a familiar Unix-like userland, which isn't surprising as much of it has been taken from NetBSD in recent



Minix 3 is the first release to support ARM chips, and specifically BeagleBoards (image: Mapper 07, Wikipedia).

releases. Indeed, Minix also uses NetBSD's *Pkgsrc* system, so a wide range of software is available with just a few commands. However, X wasn't working at the time of writing – the developers were still in the process of moving away from the crusty old XFree86 codebase.

Ultimately, Minix feels a lot like Slackware and Debian from the mid-90s. It's surprisingly usable and interesting to explore, and the documentation isn't bad either. It won't be challenging Linux or FreeBSD any time soon, but if you're interested in exploring alternative kernels or just want to expand your Unix knowledge, give it a go in *VirtualBox* or *Qemu*. (Note: for the former, create a virtual IDE hard drive image.)

“Minix 3 is now regarded as an alternative to Linux and the BSDs for embedded devices.”

PROJECT WEBSITE
www.minix3.org

How it works: Installing Minix3

```
Welcome to the MINIX 3 installation CD
-----
1. Regular MINIX 3
2. Regular MINIX 3 (with AHCI)
3. Edit menu option
4. Drop to boot prompt

Choose an option: RETURN for default; SPACE to stop countdown.
Option 1 will be chosen in 0 seconds.

Option: [1]:_
```

```
We'd like your feedback: http://minix3.org/community/
# Setup

Welcome to the MINIX 3 setup script. This script will guide you in setting
MINIX on your machine. Please consult the manual for detailed instructions.

Note 1: If the screen blanks, hit CTRL-F3 to select "software scrolling".
Note 2: If things go wrong then hit CTRL-C to abort and start over.
Note 3: Default answers, like [g], can simply be chosen by hitting ENTER.
Note 4: If you see a colon (:) then you should hit ENTER to continue.

--- Step 1: Select keyboard type ---
What type of keyboard do you have? You can choose one of:

abnt2      japanese      russian-cp1251  uk
dvorak     latin-america  russian-cp866   ukrainian-ko18-u
french     hungarian      russian          us-std
german     polish         scandinavian    us-swag
italian    portuguese     spanish

Keyboard type? (us-std)
```

```
Block size in kilobytes? [1]
You have selected to (re)install MINIX 3 in the partition /dev/c0d0p0.
The following subpartitions are now being created on /dev/c0d0p0:

  Root subpartition: /dev/c0d0p0s0  128 MB
  /home subpartition: /dev/c0d0p0s1  275 MB
  /usr subpartition: /dev/c0d0p0s2  rest of c0d0p0

Creating /dev/c0d0p0s0 for / ...
Creating /dev/c0d0p0s1 for /home ...
Creating /dev/c0d0p0s2 for /usr ...

--- Step 7: Wait for files to be copied ---

All files will now be copied to your hard disk. This may take a while.

Remaining: 7953 files. [i]
/mt/bin/sqllite3
/mt/bin/srccrc
/mt/bin/stat
/mt/bin/swp
/mt/bin/sun
/mt/bin/svlog
```

1 Boot You can use a real PC, but it's easier to just use *VirtualBox* or *Qemu* to boot the ISO image – and at this menu, hit Enter to boot up Minix.

2 Install Log in as root and enter setup to begin the process. You'll be prompted for your keyboard layout, and then move on to drive partitioning.

3 Wait If you can devote the whole drive to Minix, use the automatic mode. The files will be copied over, and then you can reboot into your new Minix installation.

Old school terminal emulator

Cool-retro-term

Hollywood typically depicts “hackers” as bespectacled geeks sitting in front of green text terminals with all sorts of incomprehensible gobbledygook scrolling by. They’ll use these cliches even in films set in the current decade, despite how daft they are. But go back to the 70s and 80s and you would actually find these flickery, eye-strain-inducing displays hooked up to mainframes and minicomputers.

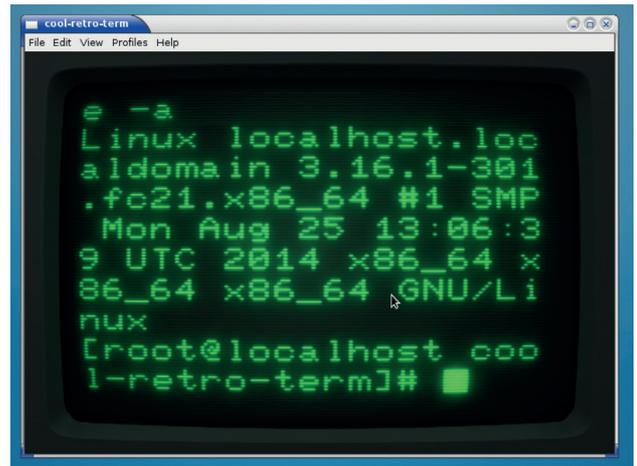
If you want something today that really harks back to the good old days of Unix, try *Cool-retro-term*. Its name says it all: it recreates an old-school terminal on your desktop, and it’s pretty cool.

Well, apart from compiling it. The developers have provided a helpful list of dependencies on the project’s website, but if you’re not running KDE you’ll be pulling in a giant

bunch of extra packages. The interface is built around *Qt 5.2*, so you’ll need an up-to-date distro to build the source code – we used a test release of Fedora 21. We followed the instructions to the letter, but we still had to install some extras manually.

Anyway, once it’s running, it’s pretty awesome. The developers have done a great job of recreating an old-school text terminal, beyond just making everything pixelated to the extreme. There’s a fuzziness around the characters, irregularity in the lighting, twitches from the virtual CRT, and warping at the edges (simulating the bulge of a non-flat screen). Even if you never used a Unix box in the 70s, it may still remind you of 8-bit computers hooked up to 80s TV sets.

Unfortunately, we couldn’t access any of the settings; the menus



Top tip: enable this full-screen on your laptop, go into a coffee shop, and watch as everyone assumes you’re the l33t3st h4x0r in the world.

simply didn’t work. It’s not clear whether this is due to a bug in the program or the mid-development status of Fedora 21, but we couldn’t explore some of the extra features. Still: you can normally customise the font size and colour scheme, and switch into a full-screen mode.

PROJECT WEBSITE
<https://github.com/Swordfish90/cool-retro-term>

Screenshot taker

Maim

One of the best command line tools for taking screenshots is *Scrot*. With a command line tool you can take batches of screenshots, selecting specific areas, which is very handy when you need to do repetitive jobs (such as collecting images for software documentation). *Maim* claims improve on *Scrot* with some features that the former lacks.

The main dependencies are **lmlib2**, **libXrandr** and **libXfixes**; these are available as **libimlib2-dev**, **libxrandr-dev** and **libxfixes-dev** on Debian-based distributions. With those in place, grab the latest source code and build it like so:

```
git clone https://github.com/naelstrof/maim.git
cd maim
cmake .
make && sudo make install
```

You can use the program straight away by entering, for example, **maim foo.png** to save the whole screen as **foo.png**. You’ll probably want to get the terminal window out of the way first, though, so add **-d** followed by a number to delay the screenshot-taking process for the specified number of seconds.

If you want to grab a part of the screen, you can specify coordinates using the **-x**, **-y**, **-h** and **-w** flags; in many cases, though, you’ll want to select an area yourself. *Maim* can’t do this on its own, but if you install *Slop* (linked to on the project’s website) then the **-s** flag lets you either click and drag to select an area, or click on a window’s titlebar to take a screenshot of just that window. The selected window is highlighted with a grey border, which you can customise using



Maim makes off-screen areas transparent, which is useful if you have multiple monitors with different resolutions.

extra settings. It’s also possible to select a specific window using the **-i** flag. This requires the window ID, which you can get by running **xdotool selectwindow**, clicking on the window you want to capture, and noting the number. Enter **maim --help** for the full list of options.

PROJECT WEBSITE
<https://github.com/naelstrof/maim>

Atari-like virtual machine

AranyM 0.9.16

There are many Atari ST/TT/Falcon emulators doing the rounds, and most of them have very good compatibility with the original machines. *AranyM* (a contraction of “Atari running on any machine”) is slightly different in that it doesn’t emulate a specific model from the ST series; it just emulates the best hardware suitable for an open source version of TOS, the ST’s old operating system.

After you’ve installed *AranyM*, you’ll also need to get a TOS image from <http://emutos.sf.net>. Like the original TOS, EmuTOS is a single-tasking operating system with certain limits, but it’s completely open source and works really well with *AranyM*. Fire up *AranyM* for the first time, and then close it; you’ll now have a file called `.aranyM/config` in your home directory. Edit this to point the EmuTOS line to the

location of the file you downloaded (eg `/home/mike/etos512k.img`) and start *AranyM* again. All being well, you’ll arrive at the desktop.

From here, you can go about setting up your virtual Atari as you wish. You can reedit the configuration file to enable hard drives using disk images or directories, and also tweak the video settings as well. Note that *AranyM* completely takes over your mouse input when you’re using it, so if you need to get the cursor back for other Linuxy work, press left Shift, left Control, left Alt and Escape at the same time.

AranyM is compatible with many original ST/Falcon games and

“AranyM is a contraction of ‘Atari running on any machine’.”



AranyM features JIT (just-in-time) translation for the emulated CPU, so it runs lightning fast.

apps, and you can also enable networking and get it online. Even if you didn’t have one of the machines at the time, it’s still great fun to play around with an OS that’s half a relic of yesteryear, and half being kept alive thanks to the passion of open source hackers.

PROJECT WEBSITE
<http://aranyM.sf.net>

Text-mode Twitter client

Rainbowstream

A command line Twitter client may seem like a crazy idea, but in most cases it works perfectly well. After all, tweets are really just plain text, and if you’ve ever spent much time with *Mutt*, *WeeChat* and similar tools, you’ll know that text-mode programs can be faster and more efficient than their GUI equivalents. *Rainbowstream* is written in Python, so the quickest way to get it is:

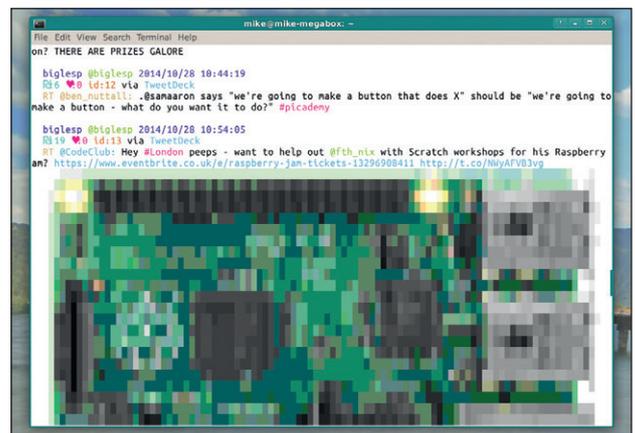
```
pip3 install rainbowstream
```

The main dependency is *Pillow*, a fork of the Python Imaging Library. Once you have it installed, enter `rainbowstream` and your web browser will pop up, prompting you to authenticate on Twitter. What’s going on here? Well, Twitter doesn’t let random applications use your account, so first it performs a check, confirming that you want to

let *Rainbowstream* read and post to your feed.

Back in the terminal, you can now begin using Twitter. Enter `h` to display a list of help topics, which you can then follow with another word – for instance, `h tweets` will show you how to compose new tweets. Entering `home` (optionally followed by a number) will show the most recent tweets on your timeline, while `mentions` will show tweets that mention you. It doesn’t take long to master, and if you spend a lot of time on Twitter, you’ll appreciate its speed and efficiency.

Certain features aren’t enabled by default, such as the ability to display inline images, as shown in our screenshot. This is really more of a novelty than anything else, as the images are inevitably low-resolution given the constraints of the



If you guessed this is a Raspberry Pi, congratulations – you can now `apt-get remove` your X server.

terminal, but it’s a lark nonetheless. Enter `config` to see a list of configuration options for *Rainbowstream*, and then enter:

```
config IMAGE_ON_TERM=True
```

Now, when you view the most recent tweets from a user (eg `view @linuxvoice`), linked images will be displayed inline.

PROJECT WEBSITE
www.rainbowstream.org

Mail client

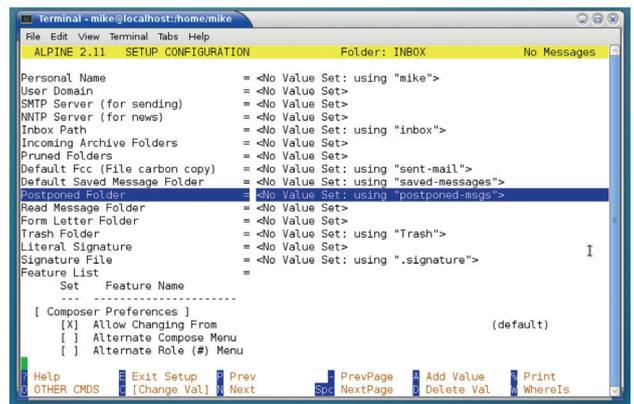
Alpine 2.11

We've given plenty of kudos to the *Mutt* email client in previous issues of Linux Voice, so some readers have asked us to also give *Alpine* a mention too. And for good reason – it's a great program, and this author used it as his primary mail client for much of the early 2000s. To explain its history, however, we need to take a deep breath: *Alpine* is a continuation of *Pine*, an email client that started life in 1989 and was influenced by *Elm* (*Electronic Mail*) before it. However, *Alpine* development ceased in 2008, and since then users have made patches to add new features, along with a fork (*re-Alpine*).

Are you following? In practical terms, it doesn't really matter: if you install *Alpine* from your distro's package repositories, you'll almost certainly get an updated version

with patches. When you start *Alpine*, you'll note that it's more welcoming than *Mutt*, with a menu-driven interface. However, like *Mutt*, there are keyboard shortcuts for various options – these are displayed along the bottom. If a shortcut looks like **^X**, that means you have to press **Ctrl+X** to activate it.

So, what makes *Alpine* great? Well, it's very fast to use: once you've learnt the keybindings, you can whizz around your mailboxes, reading, replying to and sorting mails in a fraction of the time it takes with the mouse. It's also highly customisable, and you can create colour themes for different types of messages. Almost all of



Alpine is easy and quick to set up, and doesn't require extensive hacking of config files.

the configuration can be achieved via the menus, so if you like the idea of a text-mode email client but don't want to spend hours hand-crafting configuration files, you'll love this.

Alpine supports POP, IMAP and SMTP servers out of the box, and can even browse NNTP groups. It can render HTML emails, albeit without much formatting, and supports several mailbox formats.

“Alpine is a continuation of Pine, an email client that started life in 1989.”

PROJECT WEBSITE
<http://patches.freeiz.com/alpine>

Source code analyser

PMD 5.2.0

Everyone makes mistakes, especially when you've been hacking away on the same piece of code for several hours. You may think you're working well, but niggling bugs could be creeping into your code. If you've got a deadline to reach and simply can't take a break, it's worth running a source code analyser on your work before you ship it.

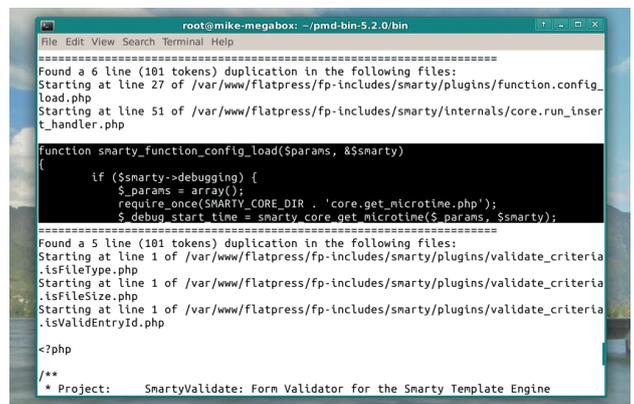
PMD is one such analyser. It aims to find flaws in your code such as unnecessary object creation, unused variables, empty catch blocks and other bloopers. Currently it only works with Java, JavaScript, XML and XSL, but an additional tool called *CPD* (the “copy-paste detector”) is provided, which identifies duplicated code in more languages including C, C++, C#, PHP and Ruby.

Both tools are written in Java, so extract **pmd-bin-5.2.0.zip** and jump into the **pmd-bin-5.2.0/bin/** directory. You'll need to provide a ruleset against which the code should be checked, along with a directory containing source code files. For instance, if you want to check some JavaScript in **/var/www/foo**, you'd use:

```
./run.sh pmd -R ecma-script-basic -d /var/www/foo
```

Many other rulesets are available – see the project's website for a full list. *PMD* will spit out its findings to **stdout**, or you can redirect them to another file by adding **> list.txt** to the end of the command.

To run *CPD*, you need to specify the language it should check, along with a location and the minimum token length, which should be reported as a duplicate. So you'd



Here's *CPD* in action, spotting repeating sections of PHP code.

use something like this:

```
./run.sh cpd --minimum-tokens 100 --files /var/www/foo --language php
```

Ideally, this should help you to reduce duplication, and stop bugs from being repeated across other areas of the program. Plugins are available to integrate *PMD* and *CPD* with various IDEs, including *Eclipse*, *NetBeans* and *JBuilder*.

PROJECT WEBSITE
<http://pmd.sourceforge.net>

FOSSPICKS Brain Relaxers

Flight simulator

FlightGear 3.2

How many games can boast a 60,000 word manual? Fortunately, you don't need to read it all to play *FlightGear*, but if you want to get the most out of the simulator, you'll spend a lot of time with it. *FlightGear* is serious business: it has been in development for nearly 20 years, it's incredibly detailed, and it has realistic physics. Well, apart from the crashes...

It's also huge. You're looking at around 2GB for the complete game, which includes over 30 aircraft and hundreds of airports around the world. Getting it is a bit difficult, as there's no standard bundle for all Linux distros, so you're left to your own distro's repositories or third-party repos.

If you're running an Ubuntu variant, there's a PPA that has this latest *FlightGear* release ([ppa:saiarcot895/flightgear](https://ppa.launchpad.net/saiarcot895/flightgear)).

If you've ever tried a complex flight sim before, you might've given up before even getting off the ground, thanks to the sheer number of controls involved. *FlightGear* is no different, but at least with the smaller aircraft like the Cessnas, you can enjoy some simple flying fun without burying your head in the docs. Hit S to start your engine, hold 9 to ramp up the throttle, and when you've got some speed, tap 8 to lift off. Then use the O and Enter keys to control the rudder.

You won't get far this way, as there are many more controls to master, but at least you'll have a bit of fun before crashing. (And really,



"Hello, this is your captain speaking. We regret to announce some minor technical issues with this Ryanair flight to Málaga. Normal service will be resumed soon."

the crashes aren't very realistic – you just bounce around on the ground.) Still, if you manage to take off and land a 777, write in and let us know...

PROJECT WEBSITE
www.flightgear.org

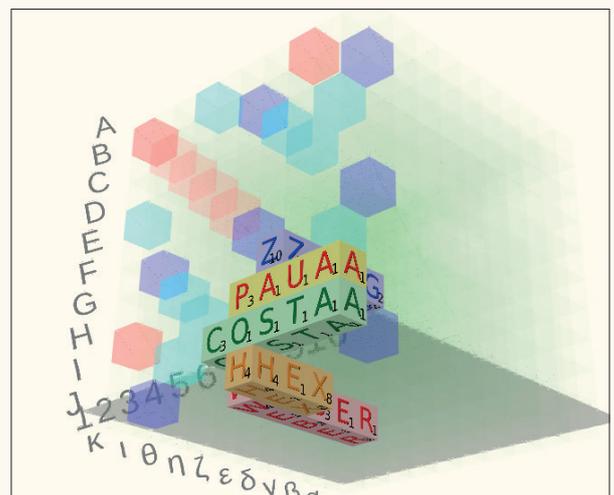
Multi-dimensional Scrabble game

Scrabble3D

Scrabble, like most popular board games, has been done to death now. Sure, it's still a great game, but from the squillions of computer versions we've seen over the years, there's little to differentiate them. Until we came across Scrabble3D, that is. This adds another dimension to the proceedings, and thereby a whole other level to the gameplay. The program is written using the *Lazarus* Pascal IDE, which means that packages are available with both *GTK* and *Qt* interfaces. The 64-bit Deb file worked fine on our Ubuntu 13.10 test machine; if you're running an RPM-based distro such as Fedora, you'll also find suitable packages at the project's website.

In its default mode, *Scrabble3D* lets you play a regular game with two, three or four players, dragging tiles from the top-right of the window onto the board on the left. The game asks for your language when you first start it, so that it can download the relevant dictionary, and you can change this later on. If you've got bored with the regular Scrabble rules, some alternatives are available: Clabbers (which lets you place anagrams of words) and Cambio Secco (whereby you can exchange all of your tiles, once per match, without losing your turn).

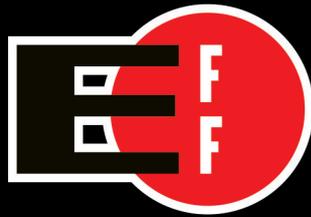
To find the 3D mode, go into Configuration > Settings, choose Advanced mode, go to Board > Configuration, and select 3D. After starting a new game, you can use



Click Game > Run demo to see the CPU play against itself in the 3D mode.

the slider under the 3D view on the right to select a plane before dropping a tile. It's a bit fiddly at first, but the manual on the project's site explains it well. 

PROJECT WEBSITE
<http://scrabble.sourceforge.net>



ELECTRONIC FRONTIER FOUNDATION

Help EFF Defend Your Rights in the Digital World eff.org/join



LISTEN TO THE PODCAST

LINUX VOICE

WWW.LINUXVOICE.COM



TUTORIALS

Dip your toe into a pool full of Linux knowledge with eight tutorials lovingly crafted to expand your Linux consciousness



Ben Everard

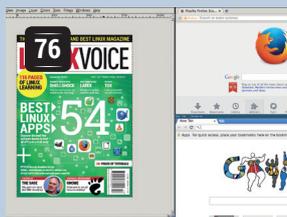
is using technology as an excuse not to venture out into the cold, dark world.

It's this time of year that technology seems most important to me. During the long days of summer, it sometimes feels like the solution to life's problems lies away from my desk and outside in the fresh air. Then October comes like a hard dose of reality with its cold winds and rain, and by November I'm ready to swear off nature for good and retreat to the comforting glow of an LCD screen. This time of year I like to pick a project to see me through the long nights until spring comes around again and brings with it the desire to step back out into the world.

This does, of course, beg the question of what technology to focus on this winter (last year was Python and the Raspberry Pi – and launching a magazine of course). I haven't quite decided yet, but I think it might be time to get my Arduino out of its summer storage and start putting it to use again. Perhaps I should just buy a load of components and start experimenting. Hmm, does the cat need an automatic cat flap? Is the temperature in our house monitored as accurately as it should be? Will this be the year I finally get round to setting up a persistence of vision system on my bike wheels? Oh, there's too much tech to play with, and not enough time. Why did I ever leave the house in summer?

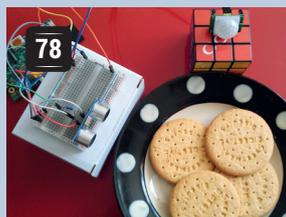
ben@linuxvoice.com

In this issue...



i3

Ditch your mouse and switch to a keyboard-driven desktop. **Ben Everard** has, and it's made him more productive than ever.



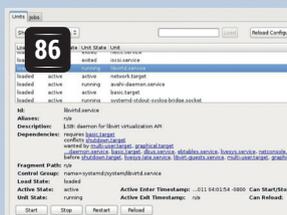
RPi Input

Les Pounder takes a Raspberry Pi, a PIR, an ultrasonic distance sensor and a reed switch and builds a biscuit protector.



Markdown

There's no need for fancy word processors: you just need a text editor and a bit of Markdown. **Marco Fioretti** explains all.



Systemd

It's the most controversial program in Linux, but how do you get it to work? **Mike Saunders** braves a flamewar by investigating.



Grub

Don't wait until your bootloader breaks before you learn how it works. Follow **Mayank Shama's** guide today.



Atlas

Learn to program one of the world's first transistor-powered supercomputers with **Juliet Kemp's** hands-on guide.

PROGRAMMING

Python

98 In part two of this series, we delve deeper into the world of fractals. Depending on your perspective these are either complex mathematical constructs that reveal hidden truths about nature, or pretty pictures. Either way, they're a great way of exploring programming concepts with Python.

Stdin and Stdout

102 These two concepts are key to all Unix-based systems, and you use them every time you use Linux (even if you don't know you do); but how do we write software that interacts with them? In this Code Ninja, we use Python to recreate a few classic command line tools and learn about Stdin and Stdout as we go.

Battle robots

104 Are you the best programmer around? Can you prove it? No, not by beating a benchmark, but in the white-hot heat of warfare! We introduce *Robocode*, programmable tank warfare, and challenge you to see who can write the most powerful tank. (No penguins were harmed in the writing of this article).

BEN EVERARD

WHY DO THIS?

- Work more efficiently.
- Keep carpal tunnel syndrome at bay.
- Get value for money from your high-resolution monitor.

i3: TILING WINDOW MANAGEMENT

Forget about touchscreens for a moment, put the mouse down and control the desktop with your keyboard.

If you've always used a stacking window manager (one in which windows overlap each other – most window managers are stacking), then the concept of a tiling window manager may seem a little strange. Using a tiling WM, you don't have much control over window placement, and there's usually no taskbar for minimised windows. Instead the whole thing is driven by keyboard commands. At first, this can seem archaic. When you first use a tiling window manager, you'll probably find that it doesn't make good use of

screen space, but this is just because you haven't learned to use it well yet.

Once you've got used to the system, you should find that you can do all your window management tasks without your fingers ever leaving the keyboard. This is much faster than flicking back and forward to the mouse. Don't worry though, you won't have to abandon your computer's rodent – you'll still be able to use it for graphical applications. Now, let's get stuck in so you can see what it's like for yourself.

Step by step: Get working with i3

1 Getting started

You can install multiple window managers on a single distro, so you can try i3 without losing your current setup. All you have to do is grab the package through your package manager (usually called **i3**). It's available in all major distros. If you're using Debian or Ubuntu (or a derivative of these), you can adjust your repositories to get the latest version of i3 by following the guide at <https://i3wm.org/docs/repositories.html>; however, this isn't necessary if you just want to try it out.

Once you've got everything installed, you just need to switch desktops. Log out of your current session and you should be able to select i3 as an option on the login screen. This will take you into i3. You should see a desktop wallpaper with a black bar along the bottom. There's no graphical menu (or anything else you might click with your mouse) as i3 is primarily keyboard-driven. If you're used to doing most of your work with the mouse, it may take a little while to get used to this, but many people find that they end up preferring a keyboard-driven interface.

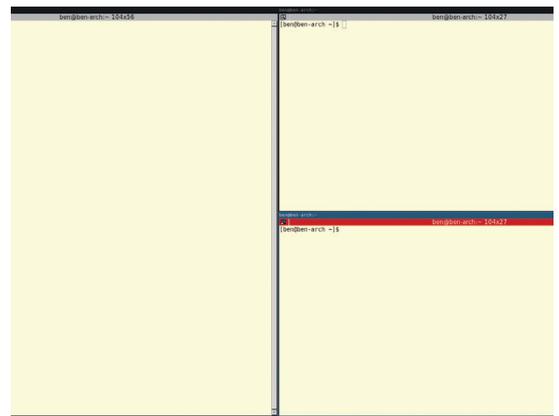


2 Windows and containers

Most of i3 is driven through a modifier key. This can be either Alt or the Windows key depending on the setup (you may have been asked to choose when starting i3). The first thing we'll do is open a terminal. This is done with Mod+Enter. Try with both Alt and Windows to see which way you have it set up.

You'll find that the terminal takes up the whole screen, and that there are no window decorations for resizing, moving or closing it. That's because i3 is a tiling window manager, and it handles all the placement and sizing.

To get an idea of how this works, press Mod+Enter again. You should find that i3 opens another terminal either next to, or below the previous terminal. Whenever you open a new window in i3 it splits the space of the current one to fit the new one in. If you press Mod+H before creating the new window, it will split the window horizontally, if you press Mod+V before, it will split vertically. This is the basic way of organising your windows on the screen.



3 Select the active window

With multiple windows open on the screen, you can use the mouse to select the one you want to use. However, this isn't very efficient. It's usually much better to move around using the keyboard. You can move between the different panes (known as containers), either by using Mod+arrow key or by using Mod+J, K, L or ; for left, down, up and right (the same as in the *Vi* text editor).

Once you're in a container, you can use Mod+F to make it full-screen, or Mod+Shift+<num> (where <num> is a number) to send it to another workspace. You can then go to the workspace by pressing Mod+<num>. If you do find yourself using the mouse, you can switch between workspaces by clicking on the numbers in the bottom left-hand corner of the screen. One final use of the mouse is that you can use it to resize containers by clicking and dragging the border between them.



4 Other applications

So far, we've only been playing with terminals. This is useful to get a feel of how the desktop works, but it's not very good for general use (unless, of course, you're a true Unix ninja). You can, of course, just start programs from terminals, but this means you have to have terminals spewing out the output of the graphical programs.

Although i3 doesn't have a graphical applications menu, it does have a menu. To open it, press Mod+D, then start typing the name of the program you want to start. This works a little like tab completion in the command line. Unlike a desktop environment (such as Gnome or KDE), i3 doesn't come with any specific applications. All Linux software should work with i3, but it's worth thinking about which applications fit in best with the keyboard-driven way of working. For example, the *Ranger* terminal-based file manager can be easier than a graphical tool like *Nautilus* or *Dolphin*.



5 More controls

What we've covered so far needs to become second nature to you, so now's a good time to start practising. Once you know all these key bindings automatically, you should be able to use i3 for most work. However, there are still some more advanced features that can come in handy.

Rather than tile a group of containers, you can have them stacked or tabbed to give more screen real-estate to the currently active container. You can switch between the various options with Mod+S, W or E (for stacking, tabbed or default respectively).

You can also resize containers by using Mod+R to enter resize mode. Once you've pressed that, you should see Resize Mode appear in the bottom-left corner. You can then use the arrow keys or J, K, L and ; (without Mod) to resize the current container. Once you've finished, press Escape to exit resize mode.



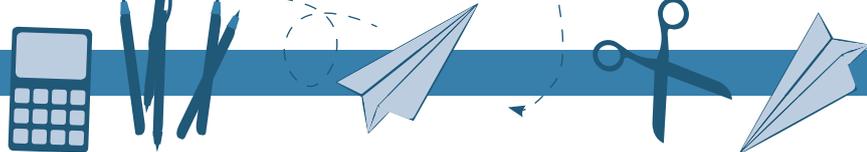
6 Configuration

Since i3 is mostly aimed at advanced computer users, it's highly customisable. This is done through modifying the config file `~/.i3/config` or `~/.config/i3/config`. In it, you should see that all the keybindings can be changed to whatever you want them to be.

As well as configuring i3, you can also get extra software to help you. *i3pystatus* (<https://github.com/enkore/i3pystatus>) is a replacement for the status bar that's extensible in Python. There's also QuickSwitch (<https://github.com/proxypoke/quickswitch-for-i3>) which is a utility to help you find and control windows in i3.

If you get stuck, i3 has good documentation at <http://i3wm.org/docs/>. There's also a stackoverflow-style FAQ at <https://faq.i3wm.org/questions> which is a good place to find solutions since most problems you face will have happened to someone before. 📖





RASPBERRY PI: SIMPLE FORMS OF INPUT

LES POUNDER

It's time to play with some affordable methods of getting input into your tiny Linux machine.

WHY DO THIS?

- Create a multi-sensor alarm system to protect coffee and biscuits.
- We will program it using Python and build a user interface using a module called easyGUI.
- Learn about sensors and alternative methods of input.

TOOLS REQUIRED

- Raspberry Pi (Any model).
- Raspbian OS.
- PIR sensor (BISS00001 are very common on eBay).
- HC-SR04 Ultrasonic Sensor (Less than £5 on eBay).
- Reed Switch (We used <http://uk.farnell.com/comus/8601-0211-015/switch-reed-spst-no-0-1a-24v-smd/dp/2409191>)
- Male to male jumpers.
- Male to female jumpers.
- 1kΩ resistor.
- Breadboard.

When thinking input methods for a computer we generally think of keyboards and mice, but there are many other different types of input. For example, the use of touch and gesture controls in mobile devices is thanks to capacitive touchscreens and accelerometers feeding data to the system that acts on the input. Sensors are unique forms of input. They provide information about the world around us and can be used to trigger alarms, gather data on animals and provide valuable data for scientific research.

In this tutorial we will look at two sensors and a magnetic switch, all of which are really simple forms of input. For our sensors, we have an ultrasonic sensor commonly used to sense distance and found in cars' parking sensors. We will then use a sensor commonly used in burglar alarms to detect movement – this is a Passive Infrared (PIR) sensor. Our magnetic switch is called a reed switch, and these are commonly used as door sensors.

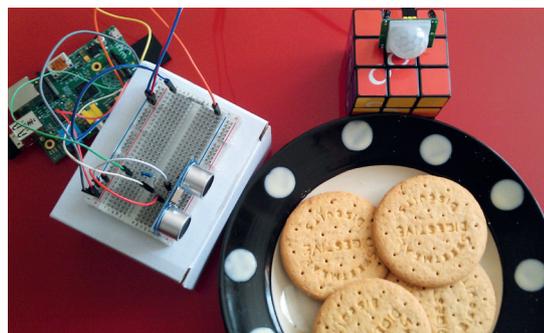
PIR sensor

Passive Infrared sensors operate by monitoring infrared light. When it detects movement, the sensor sends a high signal to your Raspberry Pi, which we have programmed to react.

PIR sensors are one of the easiest sensors to wire up to your Raspberry Pi, as they only come with three connections: VCC, which connects to the 5V pin of your GPIO (pin 2); GND or ground, which connects to pin 6; and finally Output (the pin that sends the alert signal to our Pi), which connects to pin 7 of your Pi.

With the sensors connected, let's build the Python code that will enable us to use it.

Using the PIR sensor with Python is relatively straightforward, requiring nothing more than telling the Raspberry Pi which GPIO pin the PIR sensor is attached to and to watch the status of that pin for any changes. We start by importing two modules: the first



Our final project is ready to defend against the hordes of digestive eating enemies or hungry dogs.

enables Python to use the GPIO pins. It's called **RPi.GPIO**, but this is rather unwieldy to use in our code so we rename it to **GPIO** instead:

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

```
import time
```

Next we set up the GPIO to use the logical BOARD pin numbering system and then create a variable called **PIR_PIN** to contain the real GPIO pin that we will use as an input for the trigger. The last line instructs Python that we are using pin 7 as an input and that it should expect to receive a trigger:

```
GPIO.setmode(GPIO.BOARD)
```

```
PIR_PIN = 7
```

```
GPIO.setup(PIR_PIN, GPIO.IN)
```

Now we have a little fun and print some funny system messages to the console:

```
print("Welcome to the LV Biscuit Barrier - System Loading Please Wait")
```

Installing EasyGUI

We've covered *EasyGUI* in previous issues of Linux Voice. It is the simplest method of creating a user interface, and can be inserted into existing code with relative ease.

There are two ways to install *EasyGUI*: via your package manager and via a special Python package manager.

First of all let us use the Raspbian package manager, which is called **apt**.

To install *EasyGUI* open a terminal and type in the code below followed by the enter key

```
sudo apt-get install python-easygui
```

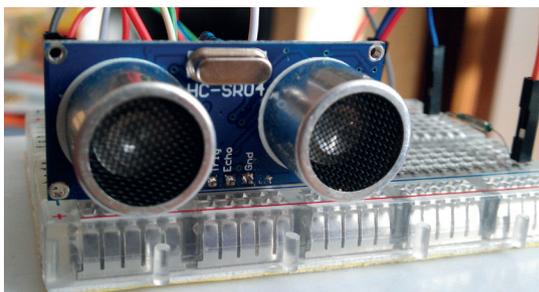
We can also use a Python package manager called *PIP* to install the packages and keep them up to date. But first we need to install *PIP* using the **apt** command.

In a terminal window type

```
sudo apt-get install python-pip
```

```
sudo pip install easygui
```

We used ultrasonic sensors to detect distance to the biscuit – they also make a great pair of eyes for a robot.



```
time.sleep(2)
```

```
print("Scanning for intruders")
```

The last segment of code is the loop that continually looks for a trigger on the **PIR_PIN**. Once someone tries to steal a biscuit they trigger the trap and a message is printed to the console informing us of the situation. With the alarm triggered, the system waits for 1 second before resetting and waiting for its next incursion:

```
while True:
```

```
    if GPIO.input(PIR_PIN):
```

```
        print("Motion Detected near the biscuits")
```

```
        time.sleep(1)
```

Reed switch

A reed switch is a small glass tube containing two strips of metal separated by about 1–2mm of space but overlapping. The switch has a “normally open” position, but when a magnet is introduced the two strips of metal snap together and complete a circuit thus allowing the current to flow through the switch. Remove the magnet and the switch opens, breaking the circuit and stopping the current flowing.

Wiring up a reed switch is extremely simple, and is very breadboard friendly. Connect the 3V3 pin from your Raspberry Pi to one end of the reed switch using a breadboard – either end of the switch can be used. The other end of the switch connects to pin 26 on your Raspberry Pi via the breadboard.

Using a reed switch with Python is just as straightforward as the PIR sensor, and we will reuse some of the same code. Let's take a look at the additions made for the reed switch.

The first part of the code remains the same:

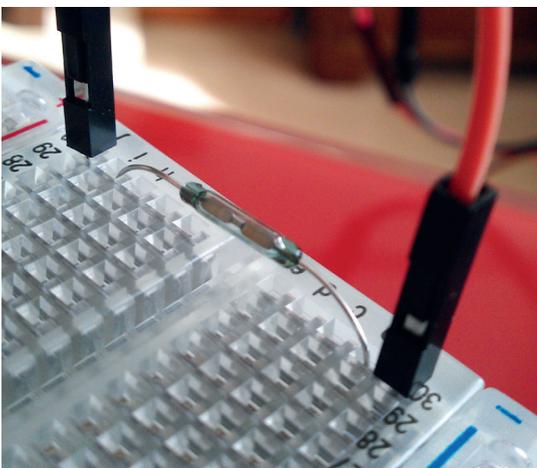
```
import RPi.GPIO as GPIO
```

```
import time
```

```
GPIO.setmode(GPIO.BOARD)
```

```
PIR_PIN = 7
```

In a similar way to the **PIR_PIN** variable, we use another variable to store the GPIO pin used for the input, which is then configured to be an input pin



A reed switch is an open switch inside a glass vial – when a magnet is introduced the switch closes.

Resistors

Resistors are an essential part of electronics, and are used to reduce the electrical current flow and in turn reduce the voltage passing through a circuit. A simple example of the use of resistors is the humble Light Emitting Diode (LED). They work with the 3.3V voltages used on the Raspberry Pi but run hot and bright, just like Rutger Hauer in Blade Runner. Using a resistor inline with the power from the Raspberry Pi pin we can reduce the current and voltage, extending the life of our LED. Without resistors,

components would have a shorter life span and we could damage our Raspberry Pi. Resistors come in a series of colours meant to identify their resistance value, measured in Ohms (Ω). Common resistors used with the Raspberry Pi are

- 220 Ω = red, red, brown, gold

- 1k Ω = brown, black, red, gold

- 10k Ω = brown, black, orange, gold

If you would like to know more about resistors, there is a great Wikipedia article <http://en.wikipedia.org/wiki/Resistor>.

ready to receive the alarm trigger:

```
reed = 26
```

```
GPIO.setup(PIR_PIN, GPIO.IN)
```

```
GPIO.setup(reed, GPIO.IN)
```

There are no changes made to our system starting/greeting message:

```
print("Welcome to the LV Biscuit Barrier - System Loading  
Please Wait")
```

```
time.sleep(2)
```

```
print("Scanning for intruders")
```

Here we see the most significant addition to our code – we create a second condition, that of the reed switch being triggered. If this condition is true then the code will print the word “Trigger” in the console:

```
while True:
```

```
    if GPIO.input(PIR_PIN) == True:
```

```
        print("Motion Detected near the biscuits")
```

```
        time.sleep(1)
```

```
    elif GPIO.input(reed) == True:
```

```
        print("Biscuit tin has been opened CODE RED!!!")
```

```
        time.sleep(1)
```

Ultrasonic sensors

Ultrasonic sensors have two ‘eyes’ – one is a trigger that sends a pulse of ultrasound towards an object, which then bounces off the object and returns to the other eye which is called echo. The distance is measured using a simple calculation:

```
Distance = Speed * Time
```

The most common ultrasonic sensor is the HC-SR04, which can be found for a few pounds on eBay. They come with 4 pins:

- **VCC** 5V power from your Raspberry Pi (Pin 1).

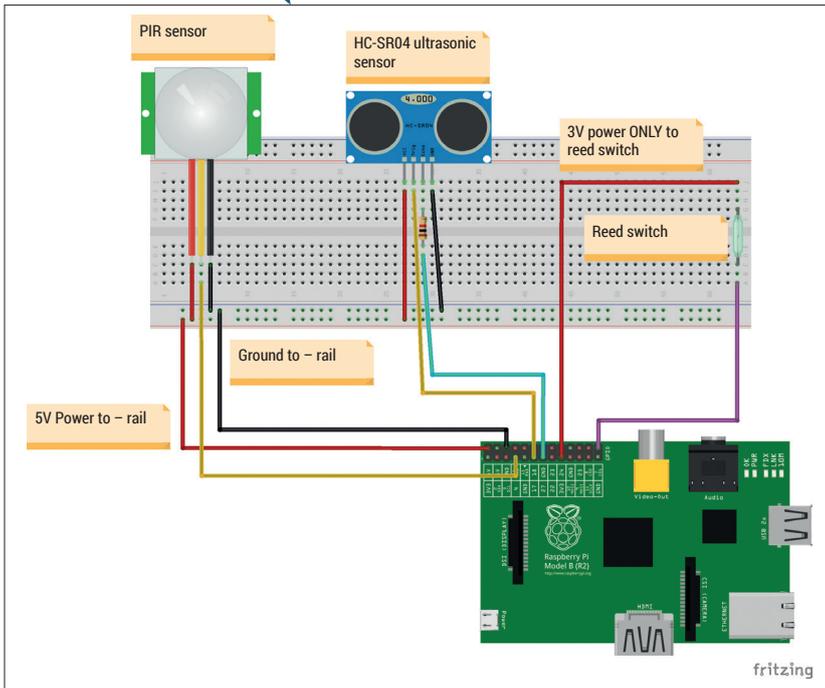
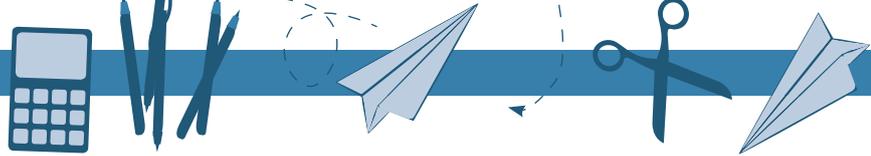
- **GND** Ground (Pin 6).

- **Trigger** Receives a signal from the Raspberry Pi to send a pulse of ultrasonic sound (Pin 11).

- **Echo** Receives the reflected ultrasonic pulse and sends a signal back to your Raspberry Pi (Pin 13).

Because we are working with 5V power we need to protect the GPIO pins of our Raspberry Pi, as they can only work with 3.3V or less. To do this we use a resistor to reduce the voltage down to something more Pi friendly. We'll use a 1k Ω resistor which has a colour code of BROWN, BLACK, RED, GOLD.

To enable the ultrasonic sensor to work with our existing Python code we need to make quite a few



Here's how the circuit is built – a larger version is available in the project's GitHub repository, see box for details.

changes. The **imports** remain the same as in previous sections, but you will notice two new variables called **trigger** and **echo**. These new variables identify the GPIO pins used to send (trigger) and receive (echo) an ultrasonic pulse. On the last line you will see something called **global distance**. This is a variable available outside and inside of a function and, without adding the **global** element, we would not be able to use the variable inside of a function that we create later in the code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
PIR_PIN = 7
reed = 26
trigger = 11
echo = 13
global distance
```

Next we configure the pins used for the ultrasonic sensor: **trigger** is an output and **echo** an input. We also instruct the **trigger** GPIO pin to be "Low" (in other words start the program with no power being sent to that pin), to reduce the chance of a false reading:

```
GPIO.setup(PIR_PIN, GPIO.IN)
GPIO.setup(reed, GPIO.IN)
GPIO.setup(trigger, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)
GPIO.output(trigger, GPIO.LOW)
```

```
We keep the system startup message the same:
print("Welcome to the LV Biscuit Barrier - System Loading
Please Wait")
time.sleep(2)
print("Scanning for intruders")
```

Next we create a function called **reading** that controls the use of the ultrasonic sensor. We reuse the **global distance** variable, thus linking the variable and enabling it to be used in our code.

Our next line is a conditional statement that checks to see if the sensor is detecting anything, if not then the code continues. To enable the sensor to 'settle' we introduce a delay using **time.sleep** of 0.3 seconds.

The next line instructs our Raspberry Pi to send an ultrasonic pulse, via sending a high signal to the trigger pin of our ultrasonic sensor. We then delay for 10 microseconds, which is just enough time for a pulse of significant length to be sent. Then we turn off the trigger pin and flow into two **while** statements.

While the echo pin is not receiving an ultrasonic pulse it updates the variable **signaloff** with the current time, and a similar construction is used for **signalon** when we receive the ultrasonic echo pulse.

With the two times recorded we now do a little maths. Subtracting the **signaloff** time from **signalon** time gives us the time taken for the pulse to be sent and return to the ultrasonic sensor, and this is saved as the variable **timepassed**:

```
def reading(sensor):
    global distance
    if sensor == 0:
        time.sleep(0.3)
        GPIO.output(trigger, True)
        time.sleep(0.00001)
        GPIO.output(trigger, False)
        while GPIO.input(echo) == 0:
            signaloff = time.time()
        while GPIO.input(echo) == 1:
            signalon = time.time()
        timepassed = signalon - signaloff
```

Now we perform another calculation, using the school equation **distance = time * speed**. Our distance variable stores the answer to the time passed multiplied by 17,000 (the speed of sound) for a half second, so a full second is 34,029 centimetres travelled. Why a half second? Well, we halve the time taken as we need to know the distance from the object, not the time taken to get there and get back. We then print the distance in the console. Our last line in the function is the end of the **if..else** conditional logic and is used to capture any errors:

```
distance = timepassed * 17000
return distance
else:
    print "Error."
```

Now we're on to the home straight and back to the main loop of our code. We keep the first two sensors, our **PIR_PIN** and **reed** the same, and we introduce another **elif** statement that checks to see if the variable distance is less than 10cm. If so, it prints "Biscuit Thief" in the console:

```
while True:
    reading(0)
    if GPIO.input(PIR_PIN) == True:
        print("Motion Detected near the biscuits")
        time.sleep(1)
    elif GPIO.input(reed) == True:
        print("Biscuit tin has been opened CODE RED!!!")
        time.sleep(1)
```

```
elif distance < 10:
```

```
    print("Biscuit thief has struck again, deploy ill-tempered jack russell terrier")
```

So far our code is just outputting the responses to the console which is good but not great. So how can we make our project great? A great user interface will help users to quickly use the project.

So where do we need a user interface in our project? First of all a cool splash page that shows off the Linux Voice logo and what the project is all about. After that we need three dialog boxes, one for each of the inputs, that will respond to any of the triggers that may occur when our biscuit thief strikes.

Coding our splash screen

Using *EasyGUI* we need to replace the system starting text with a custom splash screen. To do that we need to create a folder called **Images** and download the Linux Voice logo (these files are included in the project files from GitHub). So in our code we first need to add one more variable in the form of:

```
logo = "/Images/masthead.gif"
```

And a list that contains the possible responses to a simple yes, no question.

```
activate = ["Yes","No"]
```

With those additions made, our focus turns to the welcome message that we earlier coded:

```
print("Welcome to the LV Biscuit Barrier - System Loading Please Wait")
```

```
time.sleep(2)
```

```
print("Scanning for intruders")
```

We can replace it with:

```
splash_title = "Linux Voice Biscuit Security System V2"
```

```
splash_msg = "Would you like to protect the biscuits?"
```

```
start = buttonbox(title=splash_title,image=logo,msg=splash_msg,choices=activate)
```

So we have a title in the form of the variable **splash_title** for our dialog box, and a question for our users in the form of **splash_msg**. The potential answers are stored in the list **activate** and this answer is saved as a variable **start**, which we will use in the next piece of code.

Our focus now shifts to line 58 of the code, which is the start of an **if...else** statement. We use the answer to the splash question to drive the activation of the project. If the user answers yes, then the main body of code is run; else if the user answers no, then the program exits:

```
Line 58
```

```
if start == "Yes":
```

```
...
```

Where can I find the completed code?

I've made the code for this project publicly available via Github. For those that are familiar with Github you can clone the repository at https://github.com/lesp/LinuxVoice_Biscuit_Security or for those unfamiliar you can download the archive as a zip file from https://github.com/lesp/LinuxVoice_Biscuit_Security/archive/master.zip



Version 2 uses *EasyGUI* to create a simple user interface that's a lot friendlier to use.

```
Line 74
```

```
else:
```

```
    print("EXIT")
```

Next we create three dialog boxes that will handle the reporting of incursions in our project. Our three triggers are a PIR sensor, a reed switch and an ultrasonic sensor, and we created three conditions in our code that look like this:

```
#First condition this handles the PIR sensor being tripped
```

```
if GPIO.input(PIR_PIN) == True:
```

```
    print("Motion Detected near the biscuits")
```

```
    time.sleep(1)
```

```
#Second condition handles the reed switch being triggered by our magnetic biscuit tin lid
```

```
elif GPIO.input(reed) == True:
```

```
    print("Biscuit tin has been opened CODE RED!!!")
```

```
    time.sleep(1)
```

```
#Our third and final condition uses the output from the ultra() function to tell us if the thief's hand is less than 10 cm away
```

```
elif distance < 10:
```

```
    print("Biscuit thief has struck again, deploy ill tempered jack russell terrier")
```

For each we used a simple print function to handle the reporting, but instead of this let's use a GUI dialog box. So for:

```
print("Motion Detected near the biscuits")
```

```
print("Biscuit tin has been opened CODE RED!!!")
```

```
print("Biscuit thief has struck again, deploy ill tempered jack russell terrier")
```

Replace with:

```
msgbox(title="Motion Detected", msg="--ALERT-- I have detected movement")
```

```
msgbox(title="Biscuit tin has been opened CODE RED!!!", msg="--ALERT-- I have detected that the tin has been opened")
```

```
msgbox(title="Hand in the biscuit tin", msg="--ALERT-- Biscuit thief has struck again, deploy ill tempered jack russell terrier")
```

We can see that the **msgbox** function has a simple syntax, and that it needs a title for the dialog box and a message to report to the user.

We have learnt how three different types of input work, how they are wired up to our Raspberry Pi and how we can create a Python program that will enable us to track down our biscuit thief. Yum! 🍪

Les Pounder is a maker and hacker specialising in the Raspberry Pi and Arduino. Les travels the UK training teachers in the new computing curriculum and Raspberry Pi.

MARKDOWN: WRITE ONCE, PUBLISH ANYWHERE

Stop wasting time with word processors and follow the Markdown way to future-proof your words.

WHY DO THIS?

- We produce more and more text every year, and never know how it will be reused.
- Even Free Software can become obsolete, but a source format as simple as Markdown will always remain usable.
- If you produce lots of text, your productivity will increase. Trust us.

You never know when you will need to republish something you wrote – so why not use a multi-output source format right from the beginning?

Markdown is just one of many plain text markup systems available as Free Software (we'll return to that definition in a moment). Writing in Markdown is simple enough to understand, but the real challenge behind Markdown is in understanding why it was created, why it can be good for you, and above all, how to change your writing and publishing habits in order to get the greatest advantage from it.

If you don't regularly create significant amounts of text of whatever nature, from poetry to company or parish newsletters, Markdown will be of little or no relevance for you. But if you do, or even if you just edit, manage and publish texts written by others, Markdown can be a real blessing, for two big reasons.

The first Markdown goal is to separate content from formatting as much as possible, to save time and concentration. After decades of word processing, and seeing too many "professionally formatted" texts looking horrible on the Web or in our smartphones, we have all learned that all too often, many of the functions in traditional office suites produce just a big waste of time and energy. We obsess ourselves with fonts, margins and similar formatting details instead of just writing clearly. Then, much of that effort goes down the drain every time somebody loads what we wrote into another program, or on a small screen.

The second, even bigger rationale for Markdown may be summarised with the slogan "Write once, publish anywhere". In other words, the simpler your initial format is, the easier it will be to reuse your

content, automating as much as possible of the work. To really understand how and why Markdown is great, we have to remember the implications of the current "What You See is What You Get" (WYSIWYG) paradigm. In order to give you WYSIWYG, modern word processors automatically add tons of data and instructions to all the files they save in their native formats, and these instructions often aren't all that portable between formats. The result is more things that could go wrong whenever something changes, more temptation to make things look "just right" manually, and more work to move from one format to the other, eg from OpenDocument to HTML or PDF.

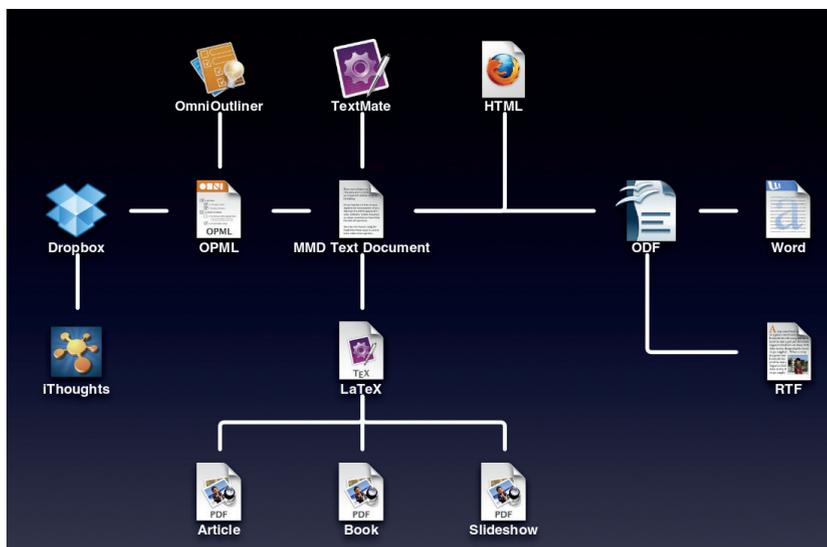
Separate style and substance

A plain text file, instead, is a file that contains only the bytes corresponding to the actual letters, punctuation and typographic "operators" – that is spaces, tabs and newlines – that we type into it using a text editor. Such files sure don't look as pretty on screen as the pages of commercial magazines, but we should learn not to care. They are extremely portable and, consequently, much more future-proof than any other alternative. They also avoid distractions: the less eye candy you can add or see, the more you are forced to ask yourself if what you wrote IS worth reading.

Very often, however, the deliberate limitations of plain text files may hurt the clarity and readability of your writing. Structures and properties like indentation, lists, typefaces or embedded images do make text easier to understand, don't they? The solution is to express them by marking the text up by adding normal characters with a special meaning, called markers or, more frequently, tags. Let's take the italic typeface as an example – a markup user would never apply it by selecting text and then clicking on some "italic" button, or menu entry. They would, instead, adopt a convention like "*//all text between a couple of slashes is meant to be italic//*" and then just type (and see on screen or paper!) those extra characters, every time italic is needed. Primitive and ugly? Maybe, but also extremely efficient.

Markdown, finally

After this long, but absolutely necessary introduction, we can finally define what Markdown is, and how to use it. First, it's a set of rules to mark up plain text, defined by John Gruber in 2004. See the "Flash introduction to Markdown syntax" box for some



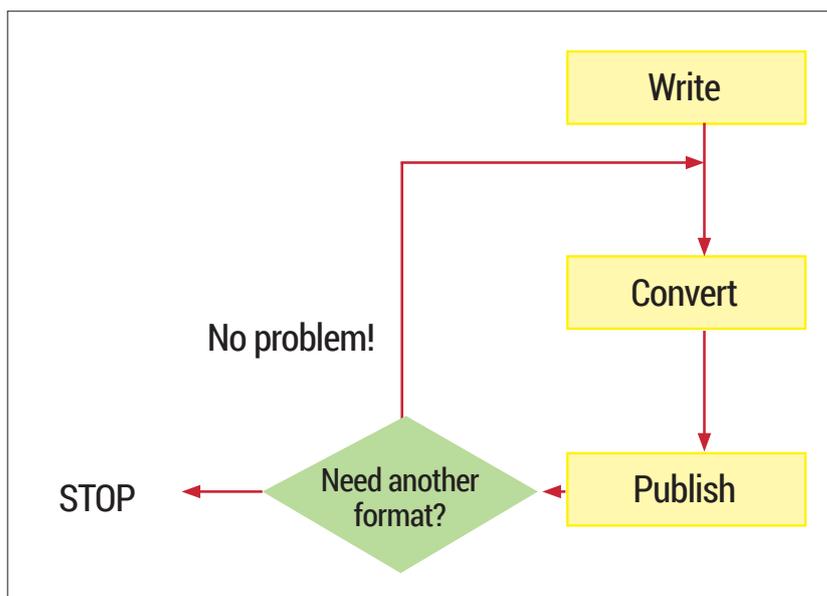
examples. Second, it's the software that converts that text to whatever target format its users need in any given moment.

The basic Markdown syntax, and all the programs written for it, were conceived for the web, to define and generate basic HTML. Very soon after the appearance of Markdown, however, other developers created syntax extensions and corresponding applications to make it possible to convert Markdown into ODF, Latex and more. Let's look at the basics first, however.

Markdown was designed to satisfy two non-negotiable criteria; efficiency and, above all, readability. If you ask your browser to show you the source of any web page, you will immediately remember that the "M" in HTML means just that: "Markup". The problem is that writing and reading raw HTML markup is tiring, error-prone and time consuming. By contrast, the main rule in Markdown is that any text formatted with it should be easily readable as-is, even by people who have never even heard of Markdown. As many advocates of this method say "the single biggest source of inspiration for Markdown's syntax is the format of plain text email".

To favour readability, all Markdown tags consist of punctuation characters, chosen to look as unintrusive as possible. The idea is to write Web-ready content faster, not to insert HTML tags faster. By deliberate choice, tags are defined only for that very small subset of HTML that corresponds to properties or operations applicable to raw text. For any other markup, from page backgrounds to navigation menus, you are supposed to use HTML (In practice, as we will see shortly, there are ways to not perform such operations manually in many cases).

In a Markdown file (the standard extension is **.md**) you can switch from Markdown to full HTML and back at any moment, without problems. The only restriction is for block-level elements – that is, practically every



long block of HTML that corresponds to one object (eg a table, or one preformatted chunk of source code), and as such is enclosed by a matching pair of tags, like `<table>` and `</table>`. Inside **.md** files, such elements must be preceded and followed by blank lines, and their enclosing tags cannot be indented with tabs or spaces.

This is the power of the Markdown flow: once you have written the source, you can repeat the conversion and publishing steps as many times you want.

The Markdown flow

In the real world, a complete Markdown-based publishing flow will have at least these three phases:

- 1 Generate your text with all the required Markdown tags.
- 2 Run the converter that generates the HTML version (or other formats, if needed) of your work.
- 3 "Publish" that version, that is send it by email to whoever may need it, put it online and so on.

Can you see the efficiency, power and flexibility, in one word: the freedom embedded in a flow like this? No? Don't worry, here it is: there is no need to perform those operations all on the same computer, all by the same person, or manually, or all at once. As far as phase 1 is concerned, any text editor, on any operating system, can be used to write Markdown (or to update Markdown files that somebody else wrote 10 years ago). And if you're using a halfway decent editor, it will surely have a Markdown syntax highlighting mode. If you wanted to publish the log files from your server, your email archive or any other body of plain text, you could write a script that converts everything to Markdown, and make the whole flow automatic.

Phases 2 and 3 can be very easily delegated to a **cron** job, and often should be. There is also no problem if years pass between one phase and the next. Everything you produce with a system like Markdown is, by definition, reusable in ways that you may not even know that you'll need one day. For example, suppose next year some blog invites you to republish stuff you wrote in 2008. If that stuff is in Markdown format, and the webmaster is willing to

LV PRO TIP

Even if you decide to not use Pandoc to generate the output formats of your Markdown documents, study and try it a few times. If nothing else, it may help you to convert all your text files to Markdown, even if they are in Microsoft or OpenDocument formats.

Documentation

Cheatsheets? Yes, a good, detailed cheatsheet is all you will need to get started with Markdown after reading this tutorial. Don't ask us for one, however. Search for them online and you'll find plenty, in all possible formats from Markdown (of course!) to desktop wallpapers. Once you have learned Markdown, study the practical YAML tutorial at <http://rnh.net/2011/01/31/yaml-tutorial>.

Before that, however, we suggest you read the post www.terminally-incoherent.com/blog/2012/05/25/Markdown-for-muggles, which is a funny encouragement to use Markdown. Another article to read before starting is 'Thoughts on Markdown' (www.leancrew.com/all-this/2010/10/thoughts-on-markdown). On a more technical level, other useful resources are the man page 'pandoc_markdown', which explains the differences between basic Markdown and its Pandoc superset, and the lists at <https://github.com/jgm/pandoc/wiki/Pandoc-vs-Multimarkdown>. That wiki page thoroughly compares the two extended converters feature by feature, starting with the input and output formats they support.

install the *Markdown QuickTags* plugin for *WordPress*, all you'll need to do is copy and paste your Markdown sources in the *WordPress* form. Please stand back one moment in silence with us, to appreciate the awesomeness of it all!

LV PRO TIP
 Whatever Markdown converter you choose, you shouldn't use it directly at the prompt. Instead, write a shell script that will call it automatically and save a log file somewhere. This will greatly reduce the possibility of mistakes, and make you work even faster.

A practical example

Here's a simple Markdown source file snippet that mixes both HTML code (in this case for a navigation menu, not the actual content!) and Markdown:

```

<!-- Navigational markup -->
<ul class="nav">
<li><a href=".">Home</a></li>
<li><a href="contact/">About</a></li>
<li><a href="contact/">Contact</a></li>
</ul>

"...MultiMarkdown provides an easy way to share formatting
between all of my devices. It's easy to learn (even for us mortals)
and immediately useful."
> --- David Sparks, [MacSparky.com](http://MacSparky.com/)

## Why Markdown and MultiMarkdown? ##

Because life is too short to waste it *formatting* text, instead of
just **writing it**.
```

The image below shows the full HTML version generated by any Markdown converter, and the image on the facing page shows the way it looks inside a web browser. See what we meant? Even without detailed explanations, or having a cheatsheet handy, both the original plain text and what is eventually shown by a browser are much more readable than the HTML.

Images and hyperlinks

One image is worth a thousand words, or so they say, but how do we handle them in plain text files? The answer is "With a little bit of care". You can tell your your Markdown converter to insert in the target file the HTML code that displays an image using this syntax:

```

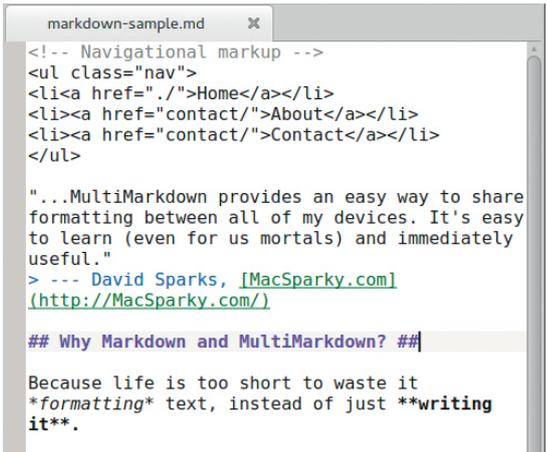
![Here you should see an image](/path/to/img.jpg "This is the image title")
```

This combines three strings: alternative text for textual browsers, a path to the image, and the image

The HTML code generated converting the Markdown source of our snippet.

```

<!-- Navigational markup -->
<ul class="nav">
<li><a href=".">Home</a></li>
<li><a href="contact/">About</a></li>
<li><a href="contact/">Contact</a></li>
</ul>
<p>"...MultiMarkdown provides an easy way to share formatting between all
of my devices. It's easy to learn (even for us mortals) and immediately
useful."</p>
<blockquote>
<p>--- David Sparks, <a href="http://MacSparky.com/">MacSparky.com</a>
</p>
</blockquote>
<h2>Why Markdown and MultiMarkdown?</h2>
<p>Because life is too short to waste it <em>formatting</em> text, instead
of just <strong>writing it</strong>.</p>
```



This is what the Markdown syntax looks like in a text editor. Marked text is coloured to make it even more readable, while embedded HTML code is left as is.

title. The problem, of course, is that this will generate enough HTML to display your image, but not enough to align, frame and size it just as you wish. There are two solutions here. One is to not use Markdown for images, but actual HTML, with all the options you may need:

```

) or one called *Dillinger* (<http://dillinger.io>), which you can even install on your own server.

On a Linux desktop, you can use the original converter, a Perl script called **markdown.pl**, or more advanced tools like Pandoc (<http://johnmacfarlane.net/pandoc>) or Multiple Markdown (MMD, <http://fletcherpenney.net/multimarkdown>). They are all command line tools, well suited for automation, and relatively simple to use. Basically, you pass them the input file (or the Standard Input) and the name of the output file in which they should save the result. The differences are in the number of input and output

#### Flash introduction to Markdown syntax

**Warning!** This is very far from being a complete description of the Markdown syntax. We only want to whet your appetite by showing how easy it is to create structured, yet highly readable plain text using Markdown. Besides, even if we had enough space, it would make no sense to give you a complete syntax primer here. The whole format is so simple, and already completely documented in countless cheatsheets that it would make no sense to copy it here:

*\*Single asterisks (or underscores) enclose italic text\**

**\*\*Couple of asterisks, instead, mean "bold!"\*\***

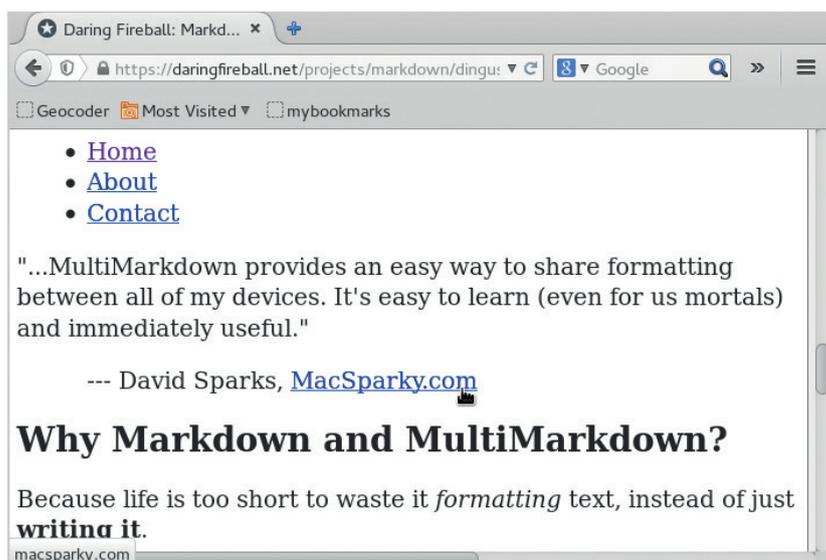
Headers can be marked in two ways. The simplest is this:

- # Level 1 header #
  - ## Level 2 header ##
- Numbered lists:

1. Foo
2. Bar

Unordered lists:

- \* first list item
  - \* another list item
- > Block quotes work just like in email.
  - > Put a ">" sign at the beginning of
  - > each of their lines.



formats supported, and in the set of Markdown tags they recognise and can process. Therefore, there is no "best converter". You must figure out by yourself which one best matches your taste or, more importantly, the type of documents you must write. The original converter, for example, only accepts basic Markdown and outputs HTML. Pandoc, instead, is a generic tool that can also be used to convert to the Markdown format Web pages or many other documents.

Pandoc defines extra Markdown tags to handle, among other things footnotes, tables, flexible ordered lists, automatic tables of contents, embedded Latex formulas, citations, and markdown inside HTML block elements. When you run the converter, multiple input files are concatenated automatically. Pandoc can even accept URLs as input files! Output goes to **stdout** by default, except for complex formats like OpenDocument or ePUB. In this way, it's also possible to generate PDF files directly from Markdown.

The other most useful features of Pandoc are the command line options that tell it to place the content of external files, for example SEO keywords, in the header of the HTML output, or at the end of a page.

MMD is another pair of Markdown syntax superset and associated converter. It is optimised for handling, among other things, tables, footnotes, citations, internal cross-references and equations. Cross-references, for example, work in this way:

```
[This string will point to an internal link][mylink]
This is where I will end up when clicking on the string above
[mylink]
```

MMD can also convert Markdown sources to Latex, which is the basis for professional-quality PDFs, with its auxiliary tool **mmd2tex**. Other, extremely important output formats supported by MMD are OpenDocument and OPML, the standard used in the *Fargo 2* blogging platform.

**Marco Fioretti is a Free Software and open data campaigner who has advocated FOSS all over the world.**

The final result: fully standard code that any browser will render without problems, with a structure perfectly matching the original Markdown file.

#### LV PRO TIP

Markdown is great and may be a good reason to change text editor, if your favourite one doesn't highlight its tags properly. We suggest you try multiplatform editors if you haven't already, as they allow you always to work in the same way!

# LINUX 101: GET THE MOST OUT OF SYSTEMD

**MIKE SAUNDERS**

It's mightily controversial – but Systemd is here to stay. Learn how to use its features, and (maybe) learn to love it too...

**WHY DO THIS?**

- Understand the big changes in modern distros.
- See how Systemd replaces SysVinit.
- Get to grips with units and the new journal.

**H**ate mail, personal insults, death threats – Lennart Poettering, the author of Systemd, is used to receiving these. The Red Hat employee recently ranted on Google+ about the nature of the FOSS community (<http://tinyurl.com/poorlennart>), lamenting that it's "quite a sick place to be in". In particular, he points to Linus Torvalds's highly acerbic mailing list posts, and accuses the kernel head honcho of setting the tone of online discussion, making personal attacks and derogatory comments the norm.

But why has Poettering received so much hate? Why does a man who simply develops open source software have to tolerate this amount of anger? Well, the answer lies in the importance of his software. Systemd is the first thing launched by the Linux kernel on most distributions now, and it serves many roles. It starts system services, handles logins, executes tasks

at specified intervals, and much more. It's growing all the time, and becoming something of a "base system" for Linux – providing all the plumbing tools needed to boot and maintain a distro.

Now, Systemd is controversial for various reasons: it eschews some established Unix conventions, such as plain text log files. It's seen as a "monolithic" project trying to take over everything else. And it's a major change to the underpinnings of our OS. Yet almost every major distribution has adopted it (or is about to), so it's here to stay. And there are benefits: faster booting, easier management of services that depend on one another, and powerful and secure logging facilities too.

So in this tutorial we'll explore Systemd's features, and show you how to get the most out of them. Even if you're not a fan of the software right now, hopefully at least you'll feel more comfortable with it by the end.

## 1 BOOTING AND SERVICES

This tongue-in-cheek animation at <http://tinyurl.com/m2e7mv8> portrays Systemd as a rabid animal eating everything in its path. Most critics haven't been so fluffy.

Almost every major distro has either adopted Systemd, or will do so in the next release (Debian and Ubuntu). In this tutorial we're using a pre-release of Fedora 21 – a distro that has been a great testing ground for Systemd – but the commands and notes

should be the same regardless of your distro. That's one of the plus points of Systemd: it obviates many of the tiny, niggling differences between distros.

In a terminal, enter **ps ax | grep systemd** and look at the first line. The **1** means that it's process ID 1, ie the first thing launched by the Linux kernel. So, once the kernel has done its work detecting hardware and organising memory, it launches the **/usr/lib/systemd/systemd** executable, which then launches other programs in turn. (In pre-Systemd days, the kernel would launch **/sbin/init**, which would then launch various other essential boot scripts, in a system known as SysVinit.)

### Taking control

Central to Systemd is the concept of units. These are configuration files with information about services (programs running in the background), devices, mount points, timers and other aspects of the operating system. One of Systemd's goals is to ease and simplify the interaction between these, so if you have a certain program that needs to start when a certain mount point is created when a certain device gets plugged in, it should be considerably easier to make all this work. (In pre-Systemd days, hacking all this together with scripts could get very ugly.) To list all units on your Linux installation, enter:



## Timer units: replacing Cron

Beyond system initialisation and service management, Systemd has its fingers in various other pies too. Notably, it can perform the job of **cron**, arguably with more flexibility (and an easier to read syntax). **Cron** is the program that performs jobs at regular intervals – such as cleaning up temporary files, refreshing caches and so forth.

If you look inside the `/usr/lib/systemd/system` directory again, you'll see that various **.timer** files are provided. Have a look at some of them with **less**, and you'll note that they follow a similar structure to the **.service** and **.target** files. The difference, however, lies in the **[Timer]** section. Consider this example:

```
[Timer]
OnBootSec=1h
OnUnitActiveSec=1w
```

Here, the **OnBootSec** option tells Systemd to activate the unit 1 hour after the system has booted. Then the second option means: activate the unit once a week after that. There's a huge amount of flexibility in the times that you can set – enter **man systemd.time** for a full list.

By default, Systemd's accuracy for timing is one minute. In other words, it will activate the unit within a minute of the time you specify, but not necessarily to the exact second. This is done for power management reasons, but if you need a timer to be executed without any delay, right down to the microsecond, you can add this line:

```
AccuracySec=1us
```

Also, the **WakeSystem** option (which can be set to true or false) defines whether or not the timer should wake up the machine if it's in suspend mode.

### LV PRO TIP

By default, **systemctl** assumes that you're referring to services when issuing commands, so you can omit the **.service** bit in most cases. For instance, instead of entering **systemctl status gdm.service** you can just use **systemctl status gdm**. The same applies to stopping and starting services.

### systemctl list-unit-files

Now, **systemctl** is the main tool for interacting with Systemd, and it has many options. Here, in the unit list, you'll notice that there's some formatting: enabled units are shown in green, and disabled are shown in red. Units marked as "static" can't be started directly – they're dependencies of other units. To narrow down the list to just services, use:

### systemctl list-unit-files --type=service

Note that "enabled" doesn't necessarily mean that a service is running; just that it can be turned on. To get information about a specific service, for instance GDM (the GNOME Display Manager), enter:

### systemctl status gdm.service

This provides lots of useful information: a human-readable description of the service, the location of the unit configuration file, when it was started, its PID, and the CGroups to which it belongs (these limit resource consumption for groups of processes).

If you look at the unit config file in `/usr/lib/systemd/system/gdm.service`, you'll see various options, including the binary to be started (ExecStart), what it conflicts with (ie which units can't be active at the same time), and what needs to be started before this unit can be activated (the "After" line). Some

units have additional dependency options, such as "Requires" (mandatory dependencies) and "Wants" (optional).

Another interesting option here is:

### Alias=display-manager.service

When you activate **gdm.service**, you will also be able to view its status using **systemctl status display-manager.service**. This is useful when you know there's a display manager running, and you want

**"Systemd eschews some established Unix conventions, such as plain text log files."**

```
Terminal - mike@localhost:/home/mike
File Edit View Terminal Tabs Help
alsa-store.service static
anaconda-direct.service static
anaconda-noshell.service static
anaconda-shell@.service static
anaconda-sshd.service static
anaconda-tmux@.service static
anaconda.service static
arp-ethers.service disabled
atd.service enabled
auditd.service enabled
autovt@.service disabled
avahi-daemon.service enabled
blk-availability.service disabled
bluetooth.service enabled
brltty.service enabled
canberra-system-bootup.service disabled
[root@localhost mike]# systemctl status atd.service
● atd.service - Job spooling tools
 Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled)
 Active: active (running) since Fri 2014-10-24 16:28:59 CEST; 1h 1min ago
 Main PID: 456 (atd)
 CGroup: /system.slice/atd.service
 └─456 /usr/sbin/atd -f
[root@localhost mike]#
```

Use **systemctl status**, followed by a unit name, to see what's going on with a service.

```

Terminal - mike@localhost:usr/lib/systemd/system
File Edit View Terminal Tabs Help
systemd is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or
(at your option) any later version.

[Unit]
Description=Journal Service
Documentation=man:systemd-journald.service(8) man:journald.conf(5)
DefaultDependencies=no
Requires=systemd-journald.socket
After=systemd-journald.socket systemd-journald-dev-log.socket syslog.socket
Before=sysinit.target

[Service]
Sockets=systemd-journald.socket systemd-journald-dev-log.socket
ExecStart=/usr/lib/systemd/systemd-journald
Restart=always
RestartSec=0
NotifyAccess=all
StandardOutput=null
CapabilityBoundingSet=CAP_SYS_ADMIN CAP_DAC_OVERRIDE CAP_SYS_PTRACE CAP_SYSLOG CAP_AUDIT_CONTROL CAP_CHOWN CAP_DAC_READ_SEARCH CAP_FOWNER CAP_SETUID CAP_SETGID
WatchdogSec=1min

```

The unit configuration files might look foreign compared to traditional scripts, but they're not hard to grasp.

to do something with it, but you don't care whether it's GDM, KDM, XDM or any of the others.

### Target locked

If you enter **ls** in the `/usr/lib/systemd/system` directory, you'll also see various files that end in `.target`. A target is a way of grouping units together so that they're started at the same time. For instance, in most Unix-like OSes there's a state of the system called "multi-user", which means that the system has booted correctly, background services are running, and it's ready for one or more users to log in and do their work – at least, in text mode. (Other states include single-user, for doing administration work, or reboot, for when the machine is shutting down.)

If you look inside `multi-user.target`, you may be expecting to see a list of units that should be active in this state. But you'll notice that the file is pretty bare – instead, individual services make themselves

dependencies of the target via the **WantedBy** option. So if you look inside `avahi-daemon.service`, `NetworkManager.service` and many other `.service` files, you'll see this line in the Install section:

**WantedBy=multi-user.target**

So, switching to the multi-user target will enable those units that contain the above line. Other targets are available (such as `emergency.target` for an emergency shell, or `halt.target` for when the machine shuts down), and you can easily switch between them like so:

**systemctl isolate emergency.target**

In many ways, these are like SysVinit runlevels, with text-mode `multi-user.target` being runlevel 3, `graphical.target` being runlevel 5, `reboot.target` being runlevel 6, and so forth.

### Up and down

Now, you might be pondering: we've got this far, and yet we haven't even looked at stopping and starting services yet! But there's a reason for this. Systemd can look like a complicated beast from the outside, so it's good to have an overview of how it works before you start interacting with it. The actual commands for managing services are very simple:

**systemctl stop cups.service**

**systemctl start cups.service**

(If a unit has been disabled, you can first enable it with `systemctl enable` followed by the unit name. This places a symbolic link for the unit in the `.wants` directory of the current target, in the `/etc/systemd/system` folder.)

Two more useful commands are `systemctl restart` and `systemctl reload`, followed by unit names. The second asks the unit to reload its configuration file. Systemd is – for the most part – very well documented, so look at the manual page (`man systemctl`) for details on every command.

### LV PRO TIP

A simple way to filter and manipulate the journal using regular Unix plain text tools is to use redirection. `journalctl -b > log.txt` will place all messages from the current boot in `log.txt`, so you can `sed` and `grep` to your heart's content.

## 2 LOG FILES: SAY HELLO TO JOURNALD

The second major component of Systemd is the journal. This is a logging system, similar to syslog, but with some major differences. And if you're a fan of the Unix way, prepare for your blood to boil: it's a binary log, so you can't just parse it using your regular command line text tools. This design decision

regularly whips up heated debates on the net, but it has some benefits too. For instance, logs can be more structured, with better metadata, so it's easier to filter out information based on executable name, PID, time and so forth.

To view the journal in its entirety, enter:

**journalctl**

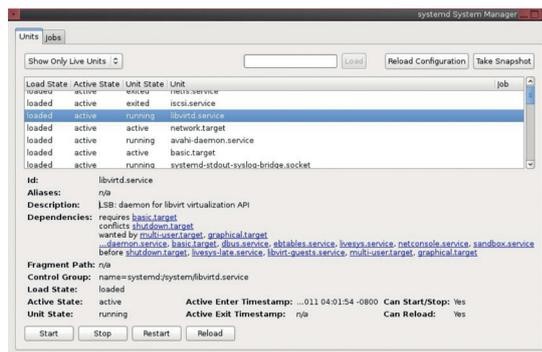
As with many other Systemd commands, this pipes the output into the `less` program, so you can scroll down by hitting space, use `/` (forward slash) to search, and other familiar keybindings. You'll also notice a sprinkling of colour here too, with warnings and failure messages in red.

That's a lot of information; to narrow it down to the current boot, use:

**journalctl -b**

And here's where the Systemd journal starts to shine. Do you want to see all messages from the

A Systemd GUI exists, although it hasn't been actively worked on for a couple of years.



```

Terminal - mike@localhost:/usr/lib/systemd/system
File Edit View Terminal Tabs Help
Oct 24 16:28:55 localhost.localdomain kernel: Console: colour VGA+ 80x25
Oct 24 16:28:55 localhost.localdomain kernel: console [tty0] enabled
Oct 24 16:28:55 localhost.localdomain kernel: allocated 12582912 bytes of page_c
Oct 24 16:28:55 localhost.localdomain kernel: please try 'cgroup_disable=memory'
Oct 24 16:28:55 localhost.localdomain kernel: hpet clockevent registered
Oct 24 16:28:55 localhost.localdomain kernel: tsc: Fast TSC calibration failed
Oct 24 16:28:55 localhost.localdomain kernel: tsc: Unable to calibrate against P
Oct 24 16:28:55 localhost.localdomain kernel: tsc: using HPET reference calibrat
Oct 24 16:28:55 localhost.localdomain kernel: tsc: Detected 2404.994 MHz process
Oct 24 16:28:55 localhost.localdomain kernel: Calibrating delay loop (skipped),
Oct 24 16:28:55 localhost.localdomain kernel: pid_max: default: 32768 minimum: 3
Oct 24 16:28:55 localhost.localdomain kernel: ACPI: Core revision 20140424
Oct 24 16:28:55 localhost.localdomain kernel: ACPI: All ACPI Tables successfully
Oct 24 16:28:55 localhost.localdomain kernel: Security Framework initialized
Oct 24 16:28:55 localhost.localdomain kernel: SELinux: Initializing.
Oct 24 16:28:55 localhost.localdomain kernel: SELinux: Starting in permissive m
Oct 24 16:28:55 localhost.localdomain kernel: Dentry cache hash table entries: 5
Oct 24 16:28:55 localhost.localdomain kernel: Inode-cache hash table entries: 26
Oct 24 16:28:55 localhost.localdomain kernel: Mount-cache hash table entries: 81
Oct 24 16:28:55 localhost.localdomain kernel: Mountpoint cache hash table entrie
Oct 24 16:28:55 localhost.localdomain kernel: Initializing cgroup subsys memory
Oct 24 16:28:55 localhost.localdomain kernel: Initializing cgroup subsys devices
Oct 24 16:28:55 localhost.localdomain kernel: Initializing cgroup subsys freezer
lines 113-135

```

Binary logging isn't popular, but the journal has some benefits, like very easy filtering of information.

previous boot? Try `journalctl -b -1`. Or the one before that? Replace `-1` with `-2`. How about something very specific, like all messages from 24 October 2014, 16:38 onwards?"

```
journalctl -b --since="2014-10-24 16:38"
```

Even if you deplore binary logs, that's still a useful feature, and for many admins it's much easier than constructing a similar filter from regular expressions.

So we've narrowed down the log to specific times, but what about specific programs? For units, try this:

```
journalctl -u gdm.service
```

(Note: that's a good way to see the log generated by the X server.) Or how about a specific PID?

```
journalctl _PID=890
```

You can even request to just see messages from a certain executable:

```
journalctl /usr/bin/pulseaudio
```

If you want to narrow down to messages of a certain priority, use the `-p` option. With 0 this will only show emergency messages (ie it's time to start praying to **\$DEITY**), whereas 7 will show absolutely everything, including debugging messages. See the manual page (`man journalctl`) for more details on the priority levels.

It's worth noting that you can combine options as well, so to only show messages from the GDM service of priority level 3 (or lower) from the current boot, use:

```
journalctl -u gdm.service -p 3 -b
```

Finally, if you just want to have a terminal window open, constantly updating with the latest journal entries, as you'd have with the `tail` command in pre-Systemd installations, just enter `journalctl -f`.

**Miked Saunders** has a PID of `-1`, divides by zero in his sleep, and knows how to sew on a button.

#### LV PRO TIP

We've been mostly poking around inside `/usr/lib/systemd/system` in this tutorial, but you may have noticed similar files inside `/etc/systemd/system` as well. What's the difference? Well, the latter takes precedence, so if you have two unit files with the same names in both locations, the one in `/etc/systemd/system` will be used. Generally, the former directory is where installed packages place their units, while the latter is for units created by root.

## Life without Systemd?

If you simply, absolutely can't get on with Systemd, you still have a few choices among the major distributions. Most notably, Slackware, the longest-running distro, hasn't made the switch yet – but its lead developer hasn't ruled it out for the future. A few small-name distros are also holding out with SysVinit as well.

But how long will this last? Gnome is becoming increasingly dependent on Systemd, and the other major desktop environments could follow suit. This is a cause of consternation in the BSD communities, as Systemd is heavily tied to Linux kernel features, so the desktops are becoming less portable, in a way. A half-way-house solution might arrive in the form of Uselessd (<http://uselessd.darknedgy.net>), which is a stripped-down version of Systemd that purely focuses on launching and supervising processes, without consuming the whole base system.



If you don't like Sysytemd, try Gentoo, which has it as a choice of init system, but doesn't force it on its users.

# GRUB 2: HEAL YOUR BOOTLOADER

MAYANK SHARMA

There are few things as irritating as a broken bootloader. Get the best out of Grub 2 and keep it shipshape.

## WHY DO THIS?

- *Grub 2* is the most popular bootloader that's used by almost every Linux distribution.
- A bootloader is a vital piece of software, but they are susceptible to damage.
- *Grub 2* is an expansive and flexible boot loader that offers various customisable options.

**“The Grub 2 Linux bootloader is a wonderful and versatile piece of software.”**

*Boot Repair* also lets you customise *Grub 2*'s options.

The *Grub 2* Linux bootloader is a wonderful and versatile piece of software. While it isn't the only bootloader out there, it's the most popular and almost all the leading desktop distros use it. The job of the *Grub* bootloader is twofold. First, it displays a menu of all installed operating systems on a computer and invites you to pick one. Second, *Grub* loads the Linux kernel if you choose a Linux operating system from the boot menu.

As you can see, if you use Linux, you can't escape the bootloader. Yet it's one the least understood components inside a Linux distro. In this tutorial we'll familiarise you with some of *Grub 2*'s famed versatility and equip you with the skills to help yourself when you have a misbehaving bootloader.

The most important parts of *Grub 2* are a bunch of text files and a couple of scripts. The first piece to know is `/etc/default/grub`. This is the text file in which you can set the general configuration variables and other

characteristics of the *Grub 2* menu (see box titled “Common user settings”).

The other important aspect of *Grub 2* is the `/etc/grub.d` folder. All the scripts that define each menu entry are housed there. The names of these scripts must have a two-digit numeric prefix. Its purpose is to define the order in which the scripts are

executed and the order of the corresponding entries when the *Grub 2* menu is built. The `00_header` file is read first, which parses the `/etc/default/grub` configuration file. Then come the entries for the Linux kernels in the `10_linux` file. This script creates one regular and one recovery menu entry for each kernel in the default `/boot` partition.

This script is followed by others for third-party apps such as `30_os-prober` and `40_custom`. The `os-prober` script creates entries for kernels and other operating systems found on other partitions. It can recognise Linux, Windows, BSD and Mac OS X installations. If your hard disk layout is too exotic for the `os-prober` script to pick up an installed distro, you can add it to the `40_custom` file (see the “Add custom entries” box).

*Grub 2* does not require you to manually maintain your boot options' configuration file: instead it generates the `/boot/grub/grub.cfg` file with the

## Graphical boot repair

A vast majority of *Grub 2* issues can easily be resolved with the touch of a button thanks to the *Boot Repair* app. This nifty little application has an intuitive user interface and can scan and comprehend various kinds of disk layouts and partitioning schemes, and can sniff out and correctly identify operating system installations inside them. The utility works on traditional computers with a Master Boot Record (MBR) as well as the newer UEFI computers with the GUID Partition Table (GPT) layout.

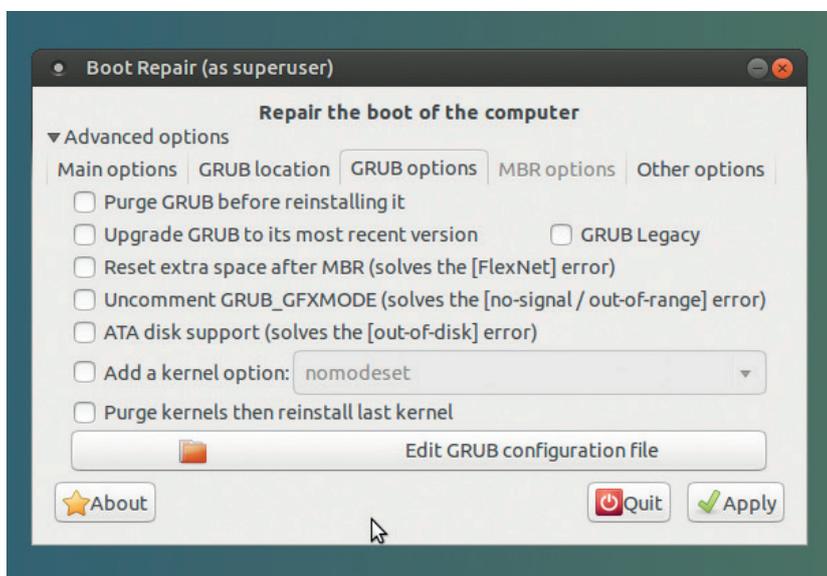
The easiest way to use *Boot Repair* is to install it inside a Live Ubuntu session. Fire up an Ubuntu Live distro on a machine with a broken bootloader and install *Boot Repair* by first adding its PPA repository with the

```
sudo add-apt-repository ppa:yannubuntu/Boot Repair
sudo apt-get update
```

before installing the app with

```
sudo apt-get install -y Boot Repair
```

Fire up the tool once it's installed. The app will scan your hard disk before displaying its interface, which is made up of a couple of buttons. To follow the advice of the tool, simply press the Recommended Repair button, which should fix most broken bootloaders. After it's restored your bootloader, the tool also spits out a small URL which you should note. The URL contains a detailed summary of your disks, including your partitions along with the contents of important *Grub 2* files including `/etc/default/grub` and `boot/grub/grub.cfg`. If the tool hasn't been able to fix your bootloader, you can share the URL on your distro's forum boards to allow others to understand your disk layout and offer suggestions.



**grub2-mkconfig** command. This utility will parse the scripts in the `/etc/grub.d` directory and the `/etc/default/grub` settings file to define your setup.

## Bootloader bailout

*Grub 2* boot problems can leave the system in several states. The text on the display where you'd expect the bootloader menu gives an indication of the current state of the system. If the system stops booting at the **grub>** prompt, it means the *Grub 2* modules were loaded but it couldn't find the **grub.cfg** file. This is the full *Grub 2* command shell and you can do quite a bit here to help yourself. If you see the **grub rescue>** prompt, it means that the bootloader couldn't find the *Grub 2* modules nor could it find any of your boot files. However, if your screen just displays the word 'GRUB', it means the bootloader has failed to find even the most basic information that's usually contained in the Master Boot Record.

You can correct these *Grub* failures either by using a live CD or from *Grub 2*'s command shell. If you're lucky and your bootloader drops you at the **grub>** prompt, you have the power of the *Grub 2* shell at your disposal to correct any errors.

The next few commands work with both **grub>** and **grub rescue>**. The **set pager=1** command invokes the pager, which prevents text from scrolling off the screen. You can also use the **ls** command which lists all partitions that *Grub* sees, like this:

```
grub> ls
(hd0) (hd0,msdos5) (hd0,msdos6) (hd1,msdos1)
```

As you can see, the command also lists the partition table scheme along with the partitions.

You can also use the **ls** command on each partition to find your root filesystem:

```
grub> ls (hd0,5)/
lost+found/ var/ etc/ media/ bin/ initrd.gz
boot/ dev/ home/ selinux/ srv/ tmp/ vmlinuz
```

You can drop the **msdos** bit from the name of the partition. Also, if you miss the trailing slash and instead say **ls (hd0,5)** you'll get information about the partition including its filesystem type, total size, and last modification time. If you have multiple partitions, read the contents of the `/etc/issue` file with the **cat** command to identify the distro, such as **cat (hd0,5)/etc/issue**.

Assuming you find the root filesystem you're looking for inside **(hd0,5)**, make sure that it contains the `/boot/grub` directory and the Linux kernel image you wish to boot into, such as **vmlinuz-3.13.0-24-generic**. Now type the following:

```
grub> set root=(hd0,5)
grub> linux /boot/vmlinuz-3.13.0-24-generic root=/dev/sda5
grub> initrd /boot/initrd.img-3.13.0-24-generic
```

The first command points *Grub* to the partition housing the distro we wish to boot into. The second command then tells *Grub* the location of the kernel image inside the partition as well as the location of the root filesystem. The final line sets the location of the initial ramdisk file. You can use tab autocompletion to

## Grub 2 and UEFI

UEFI-enabled machines (more or less, any machine sold in the last couple of years) have added another layer of complexity to debugging a broken *Grub 2* bootloader. While the procedure for restoring a *Grub 2* install on a UEFI machine isn't much different than it is on a non-UEFI machine, the newer firmware handles things differently, which results in mixed restoration results.

On a UEFI-based system, you do not install anything in the MBR. Instead you install a Linux EFI bootloader in the EFI System Partition (ESP) and set it as the EFI's default boot program using a tool such as **efibootmgr** for Linux, or **bcdedit** for Windows.

As things stand now, the *Grub 2* bootloader should be installed properly when installing any major desktop Linux distro, which will happily coexist with Windows 8. However, if you end up with a broken bootloader, you can restore the machine with a live distro. When you boot the live medium, make sure you boot it in the UEFI mode. The computer's boot menu will have two boot

options for each removable drive – a vanilla option and an option tagged with UEFI. Use the latter to expose the EFI variables in `/sys/firmware/efi/`.

From the live environment, mount the root filesystem of the broken installation as mentioned in the tutorial. You'll also have to mount the ESP partition. Assuming it's `/dev/sda1`, you can mount it with

```
sudo mount /dev/sda1 /mnt/boot/efi
```

Then load the **efivars** module with **modprobe efivars** before chrooting into the installed distribution as shown in the tutorial.

Here on, if you're using Fedora, reinstall the bootloader with the **yum reinstall grub2-efi shim** command followed by **grub2-mkconfig -o /boot/grub2/grub.cfg** to generate the new configuration file. Ubuntu users can do this with

```
apt-get install --reinstall grub-efi-amd64
```

With the bootloader in place, exit chroot, unmount all partitions and reboot to the *Grub 2* menu.

fill in the name of the kernel and the `initrd`, which will save you a lot of time and effort.

Once you've keyed these in, type **boot** at the next **grub>** prompt and *Grub* will boot into the specified operating system.

Things are a little different if you're at the **grub rescue>** prompt. Since the bootloader hasn't been able to find and load any of the required modules, you'll have to insert them manually:

```
grub rescue> set root=(hd0,5)
grub rescue> insmod (hd0,5)/boot/grub/normal.mod
grub rescue> normal
grub> insmod linux
```

As you can see, just like before, after we use the **ls** command to hunt down the Linux partition, we mark it with the **set** command. We then insert the **normal** module, which when activated will return us to the

*Grub 2* has a command line, which you can invoke by pressing **C** at the bootloader menu.

```
GNU GRUB version 1.99-27+deb7u2

Minimal BASH-like line editing is supported. For the first word,
TAB lists possible command completions. Anywhere else TAB lists
possible device or file completions. ESC at any time exits.

grub> ls
(hd0) (hd0,msdos5) (hd0,msdos1) (hd1) (hd2) (hd3)
grub> ls (hd0,msdos5)
Partition hd0,msdos5: Not a known filesystem - Partition start at
5928960 - Total size 360448 sectors
grub> ls (hd0,msdos1)
Partition hd0,msdos1: Filesystem type ext2 - Last modification time
2014-09-26 14:53:40 Friday, UUID 7a53ccc5-963f-4628-80ba-3696bbc641a9 -
Partition start at 2048 - Total size 5924864 sectors
grub> ls (hd0,msdos1)/
lost+found/ var/ etc/ media/ bin/ boot/ dev/ export/ home/ initrd.img lib/
lib64/ mnt/ opt/ proc/ root/ run/ sbin/ selinux/ srv/ sys/ tmp/ usr/ vmlinuz
grub> _
```

```

bodhi@bodhi-Lenovo-G505s: ~
File Edit View Search Terminal Help
#
DO NOT EDIT THIS FILE
#
It is automatically generated by grub-mkconfig using templates
from /etc/grub.d and settings from /etc/default/grub
#
BEGIN /etc/grub.d/00_header
if [-s $prefix/grubenv]; then
 set have_grubenv=true
 load_env
fi
if ["${next_entry}"]; then
 set default="${next_entry}"
 set next_entry=
 save_env next_entry
 set boot_once=true
else
 set default="0"
fi
if [x"${feature_menuentry_id}" = xy]; then
 menuentry_id_option="--id"
else
 menuentry_id_option=""
fi
export menuentry_id_option

if ["${prev_saved_entry}"]; then
 set saved_entry="${prev_saved_entry}"
:

```

To disable a script under the `/etc/grub.d`, all you need to do is remove the executable bit, for example with `chmod -x /etc/grub.d/20_memtest86+` which will remove the 'Memory Test' option from the menu.

standard `grub>` mode. The next command then inserts the linux module in case it hasn't been loaded. Once this module has been loaded you can proceed to point the boot loader to the kernel image and initrd files just as before and round off the procedure with the `boot` command to bring up the distro.

Once you've successfully booted into the distro, don't forget to regenerate a new configuration file for *Grub* with the

```
grub-mkconfig -o /boot/grub/grub.cfg
```

command. You'll also have to install a copy of the bootloader into the MBR with the

```
sudo grub2-install /dev/sda
```

command.

### Dude, where's my Grub?

The best thing about *Grub 2* is that you can reinstall it whenever you want. So if you lose the *Grub 2*

bootloader, say when another OS like Windows replaces it with its own bootloader, you can restore *Grub* within a few steps with the help of a live distro. Assuming

you've installed a distro on `/dev/sda5`, you can reinstall *Grub* by first creating a mount directory for the distro with

```
sudo mkdir -p /mnt/distro
```

and then mounting the partition with

```
mount /dev/sda5 /mnt/distro
```

You can then reinstall *Grub* with

```
grub2-install --root-directory=/mnt/distro /dev/sda
```

**“The best thing about Grub 2 is that you can reinstall it whenever you want.”**

This command will rewrite the MBR information on the `/dev/sda` device, point to the current Linux installation and rewrite some *Grub 2* files such as `grubenv` and `device.map`.

Another common issue pops up on computers with multiple distros. When you install a new Linux distro, its bootloader should pick up the already installed distros. In case it doesn't, just boot into the newly installed distro and run

### grub2-mkconfig

Before running the command, make sure that the root partitions of the distros missing from the boot menu are mounted. If the distro you wish to add has `/root` and `/home` on separate partitions, only mount the partition that contains `/root`, before running the `grub2-mkconfig` command.

While *Grub 2* will be able to pick most distros, trying to add a Fedora installation from within Ubuntu requires one extra step. If you've installed Fedora with its default settings, the distro's installer would have created LVM partitions. In this case, you'll first have to install the `lvm2` driver using the distro's package management system, such as with

```
sudo apt-get install lvm2
```

before *Grub 2*'s `os-prober` script can find and add Fedora to the boot menu.

### Thorough fix

If the `grub2-install` command didn't work for you, and you still can't boot into Linux, you'll need to completely reinstall and reconfigure the bootloader. For this task, we'll use the venerable `chroot` utility to change the run environment from that of the live CD to the Linux install we want to recover. You can use any Linux live CD for this purpose as long as it has the `chroot` tool. However, make sure the live medium is for the same architecture as the architecture of the installation on the hard disk. So if you wish to `chroot` to a 64-bit installation you must use an amd64 live distro.

After you've booted the live distro, the first order of business is to check the partitions on the machine. Use `fdisk -l` to list all the partitions on the disk and

### Common user settings

*Grub 2* has lots of configuration variables. Here are some of the common ones that you're most likely to modify in the `/etc/default/grub` file. The `GRUB_DEFAULT` variable specifies the default boot entry. It will accept a numeric value such as 0, which denotes the first entry, or "saved" which will point it to the selected option from the previous boot. The `GRUB_TIMEOUT` variable specifies the delay before booting the default menu entry and the `GRUB_CMDLINE_LINUX` variable lists the parameters that are passed on the kernel command line for all Linux menu entries.

If the `GRUB_DISABLE_RECOVERY` variable is set to `true`, the recovery mode menu entries will not be generated. These entries boot the distro into single-user mode from where you can repair your system with command line tools. Also useful is the `GRUB_GFXMODE` variable, which specifies the resolution of the text shown in the menu. The variable can take any value supported by your graphics card.

make note of the partition that holds the *Grub 2* installation that you want to fix.

Let's assume we wish to restore the bootloader from the distro installed in `/dev/sda5`. Fire up a terminal and mount it with:

```
sudo mount /dev/sda5 /mnt
```

Now you'll have to bind the directories that the *Grub 2* bootloader needs access to in order to detect other operating systems:

```
$ sudo mount --bind /dev /mnt/dev
```

```
$ sudo mount --bind /dev/pts /mnt/dev/pts
```

```
$ sudo mount --bind /proc /mnt/proc
```

```
$ sudo mount --bind /sys /mnt/sys
```

We're now all set to leave the live environment and enter into the distro installed inside the `/dev/sda5` partition via **chroot**:

```
$ sudo chroot /mnt /bin/bash
```

You're now all set to install, check, and update *Grub*. Just like before, use the

```
sudo grub2-install /dev/sda
```

command to reinstall the bootloader. Since the **grub2-install** command doesn't touch the **grub.cfg** file, we'll have to create it manually with

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

That should do the trick. You now have a fresh copy of *Grub 2* with a list of all the operating systems and distros installed on your machine. Before you can restart the computer, you'll have to exit the chrooted system and unmount all the partitions in the following

### Add custom entries

If you wish to add an entry to the bootloader menu, you should add a boot stanza to the **40\_custom** script. You can, for example, use it to display an entry to boot a Linux distro installed on a removable USB drive. Assuming your USB drive is **sdb1**, and the **vmlinuz** kernel image and the **initrd** files are under the root (**/**) directory, add the following to the **40\_custom** file:

```
menuentry "Linux on USB" {
 set root=(hd1,1)
 linux /vmlinuz root=/dev/sdb1 ro quiet splash
 initrd /initrd.img
}
```

```
}
}

For more accurate results, instead of device and partition names you can use their UUIDs, such as
```

```
set root=UUID=54f22dd7-eabe
```

Use

```
sudo blkid
```

to find the UUIDs of all the connected drives and partitions. You can also add entries for any distros on your disk that weren't picked up by the **os-prober** script, as long as you know where the distro's installed and the location of its kernel and **initrd** image files.

order:

```
$ exit
```

```
$ sudo umount /mnt/sys
```

```
$ sudo umount /mnt/proc
```

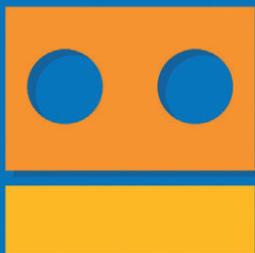
```
$ sudo umount /mnt/dev/pts
```

```
$ sudo umount /mnt/dev
```

```
$ sudo umount /mnt
```

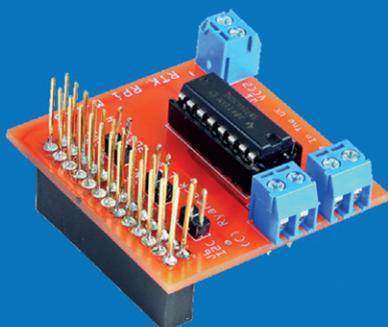
You can now safely reboot the machine, which should be back under *Grub 2*'s control, and the bootloader under yours! 🐧

Mayank Sharma has been tinkering with Linux since the 90s and contributes to a variety of technical publications on both sides of the pond.



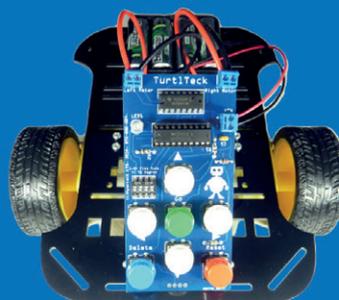
# Ryanteck LTD

## Stockist of Raspberry Pi Accessories, Sensors, Robotics & More Electronics Products

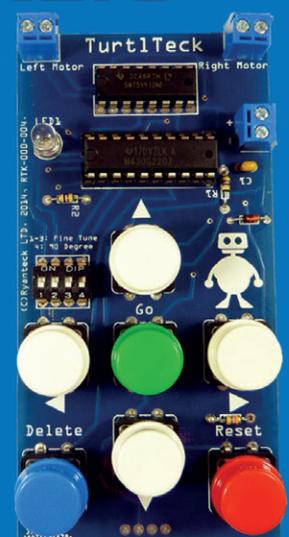


RPi Motor Controllers  
From £9.49

Raspberry Pi Robotics Kits  
From £24.99



TurtlTeck Robot Kit  
Only £29.99



TurtlTeck Robot  
Controller £16.49

<http://Store.Ryanteck.UK>

# ATLAS: THE UK'S SUPERCOMPUTER

**JULIET KEMP**

In the 1950s came the transistor, and with the transistor came the supercomputer – here's how to program one of the first.

**A**tlas, in Manchester, was one of the first supercomputers; it was said that when Atlas went down, the UK's computing capacity was reduced by half. Today supercomputers are massively parallel and run at many, many times the speed of Atlas. (The fastest in the world is currently Tianhe-2, in Guangzhou, China, running at 33 petaflops, or over a thousand million times faster than Atlas.) But some of the basics of modern computers still owe something to the decisions made by the Atlas team when they were trying to build their 'microsecond engine'.

The computers of the early 1950s were built with vacuum tubes, which made for machines which were enormous, unreliable, and very expensive. The University of Manchester computing team already had one of these, the Manchester Mark 1 (which Alan Turing worked with), which began operation in April 1949. They were working on a smaller version when Tom Kilburn, director of the group, set a couple

of his team to designing a computer which used transistors.

The result was the Transistor Computer, the world's first transistorised computer, first operational

in April 1953. It used germanium point-contact transistors, the only type available at the time, which were even less reliable than valves; but they were a lot cheaper to run, using much less power. It did still use valves for memory read/write and for the clock cycle, so it wasn't fully transistorised. (The first fully transistorised computer was the Harwell CADET, in

1955.) Once junction transistors became available, the second version of the machine was more reliable.

## Building Atlas

After the success of the Transistor Computer, the next challenge the team set themselves was to build a "microsecond engine" – a computer that could operate at one microsecond per instruction (or close to it), so managing a million instructions a second. (This is not quite the same as one megaflop, as FLOPS measure floating-point operations, not instructions, and are a little slower than instructions.)

The machine was initially called MUSE (after the Greek letter  $\mu$ , meaning one-millionth), but was renamed Atlas once the Ferranti company became involved in 1958. When Atlas was officially first commissioned, in December 1962, it was one of the most powerful computers in the world, running at (at peak) 1.59 microseconds per instruction, or around 630,000 instructions/second.

Atlas was an asynchronous processing machine, with 48-bit words, 24-bit addressing, and (in the Manchester installation) 16k word core store and 96k word drum store. It also had over a hundred index registers to use for address modification. It was fitted up for magnetic tape (a big novelty at the time and much faster than paper tape).

One important feature was instruction pipelining, which meant being able to speed up programs beyond merely running instructions more quickly. With instruction pipelining, the CPU begins to fetch the next instruction while the current one is still processing. Instead of holding up the whole CPU while a single instruction goes through the CPU's various parts, pipelining means that instructions follow one another from point A to point B to point C through the CPU, maximising the amount of work being done by the CPU at a particular time, and minimising the overall time. Obviously this does require appropriate programming to take advantage of it.

Atlas' "Extracode" setup also allowed certain more complex instructions to run as software rather than be included in the hardware. The most significant bit of the top 10 bits of a word determined whether an instruction was a normal hardware instruction, or an Extracode instruction. An Extracode instruction meant that the program would jump to what was basically a subroutine in the ROM, and run that. This was a way of reducing the complexity of the hardware while still

**"Germanium transistors were even less reliable than valves, but were a lot cheaper to run."**

The University of Manchester's ATLAS machine, photographed on 1 January 1963. Photo: Iain MacCallum



being able to provide those complicated instructions 'baked in' to the machine (and thus easy to use for programmers). Extracode instructions were used particularly for calculations like sine/cosine and square root (inefficient to wire into the hardware); but they were also used for operating system functions like printing to output or reading from a tape. (See the next section for more on the Atlas Supervisor.)

The first production Atlas, the Manchester installation, started work in 1962, although the OS software, Atlas Supervisor (see below) wasn't fully operational until early 1964. Ferranti and the University shared the available time on Atlas, running between them up to a thousand user programs in a 20-hour 'day'. The value of the machine to the University was estimated at £720,000 per year in 1969, if they'd had to buy it in externally.

## Software

A 48-bit Atlas instruction was divided into four parts: a 10-bit function code, 7-bit Ba (bits 10-16) and 7-bit Bm (bits 17-25) index registers, and a 24-bit address.

There were two basic types of instruction: B-codes, which used Bm as a modifier and Ba as a register address and did integer operations; and A-codes, which provided floating point arithmetic.

The B index registers were used to modify the given address to get the 'correct' one (useful for moving through a series of memory locations); having two index fields meant Atlas could be double-indexed. You could also test the Bm register and then do something specific with the contents of the Ba register, depending on the result of the test. Specifically, there was a general form of:

"if CONDITION then load register Ba with address N (and optionally act on Bm); otherwise do nothing"

Since register B127 was the program counter, this could be used as a program operation transfer. Simply set N to the location you want to jump to, and set Ba to B127. If the condition is true, B127 now contains address N, and the program jumps to N.

Other B registers also had specific roles, and there is a comprehensive list of these registers and many other details of the system in the short book *The Story of ATLAS* by Iain Stinson (<http://elearn.cs.man.ac.uk/~atlas/docs/london%20atlas%20book.pdf>).

Atlas Supervisor was the Atlas operating system, which managed resources and allocated them between user programs and other tasks, including managing virtual memory. It's been called "the first recognisable modern OS" in terms of how it managed jobs and resources. At any given time, multiple user programs could be running, and it was Atlas Supervisor's responsibility to manage resources and workload. The Scheduler and Job Assembler would assemble all parts of a job and sort it into one of two queues (requires its own magnetic tape, or does not). The Central Executive took care of program-switching, error-monitoring, Extracodes, and memory management. Output Assembly handled output

## Transistors

Vacuum tubes, used in all the 1940s computers, were far from ideal. Everyone in the industry was keen for something different. In particular, Bell (the telephone company) wanted a more reliable component for telephone systems. It put together a team to research transistors, based on an idea patented in the late 1920s by physicist Julius Lilienfeld. The Bell Labs team produced a working transistor in 1947 (a French team repeated this independently in 1948). Bell Labs' Shockley, Bardeen, and Brattain won the Nobel Prize in Physics in 1956 for their work.

Fundamentally, transistors control and direct the flow of electricity. They act as switches (sending current one way or another, or switching it off), and they can also amplify current, making the output power greater than the input power. The first transistors were made from germanium, which when pure is an insulator, but when

slightly impure becomes a semiconductor, which is what is needed for a transistor to work. The amount of impurity must be tightly controlled to create the correct effect. Germanium transistors were very quickly replaced by junction transistors, which are more robust and easier to make.

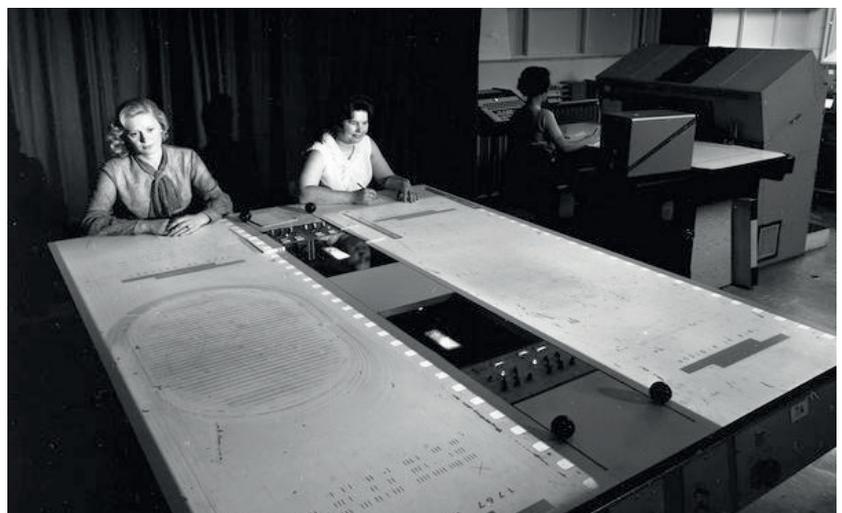
Transistors are an essential part of nearly all modern electronics, although most modern transistors are silicon rather than germanium. The fact that they can be easily mass-produced at low cost (more than ten million transistors can be made per US cent) has been a crucial part of the development of mass modern technology. These days they are usually part of an integrated circuit rather than wired together as with early transistor computers, but they're still at the core of all practical electronics. Estimates of transistors made per year vary between about half a billion and a billion per person on the Earth, and those numbers are still going up.

potentially onto many different devices, maintaining a list of documents to be output.

One of the radical innovations of Atlas was virtual memory. Computers had (and still have) two levels of memory: main (working) memory and secondary (disk; or drums/tapes back in the 1950s) memory. A program can only deal directly with main memory. For a programmer trying to perform a calculation (such as matrix multiplication) that couldn't fit into main memory, a large part of the job became working out how to switch data in and out of secondary memory, how to do it most efficiently, what blocks (pages) to divide it into, how to swap it in and out, and so on. The designers of Atlas were working programmers (as was everyone working in computers at the time), and it was very clear to them that automating this process would make programmers' lives much easier. Atlas' virtual memory had three important features:

- It translated addresses automatically into memory locations (so the programmer didn't need to keep

Ann Moffat worked with Ferranti on the Manchester Atlas from 1962, and in 1966 was one of the earliest teleworkers – here seen writing programs to analyse Concorde's black box, with her daughter. Copyright Rutherford Appleton Laboratory and the Science and Technology Facilities Council (STFC). [www.chilton-computing.org.uk](http://www.chilton-computing.org.uk).





The Atlas machine room at Chilton in 1967. Copyright Rutherford Appleton Laboratory and the Science and Technology Facilities Council (STFC). [www.chilton-computing.org.uk](http://www.chilton-computing.org.uk)

- track of memory locations by hand).
  - It had demand paging: the address translator would automatically load a required page of data into main memory when it was required.
  - It had an algorithm which identified the currently least-required pages and moved them back into secondary memory.
- Fundamentally, this is still what virtual memory does today, and it does, as expected, make programming massively more straightforward. It's also vital for running multiple programs at the same time, allowing the OS to swap parts of jobs in and out of memory as they are required.

### Emulator

An Atlas simulator is available from the Institute for Computing Systems Architecture (University of Edinburgh – [www.icsa.inf.ed.ac.uk/research/groups/hase/models/atlas/index.html](http://www.icsa.inf.ed.ac.uk/research/groups/hase/models/atlas/index.html)). You can download their three sample programs from their website. This is a simulator rather than an emulator, in that it simulates the operation of the Atlas architecture by modelling its internal state, but doesn't pretend to give the experience of operating the whole machine.

To run the simulator, you'll first need to download and install HASE III. There are detailed instructions [www.icsa.inf.ed.ac.uk/research/groups/hase/models/use.html](http://www.icsa.inf.ed.ac.uk/research/groups/hase/models/use.html), but basically you download the **jar** file, then type:

```
java -jar Setup_HASE_3.5.jar
```

at a terminal window. Run as root to be able to install for all users of the machine, or as a user to install for just that user. You can then run the **bin/Hase** executable from wherever you installed the program.

To run one of the Atlas projects, download one of the samples and unzip it, choose Open Project from the HASE menu, then choose the relevant **.edl** file. So for Atlas\_V1.3, the project that demonstrates each of the various instructions, choose **V1.3/atlas\_v1.3.edl**.

To compile the project, first, if you installed HASE as root, you'll need to make sure that the user as which you're running has write access to **hase/hase-iii/lib**. (This seems only to be necessary for the first

compile.) Next, go to Project > Properties > Compiler, and make sure that the **Hase** directory is set correctly to where you installed HASE. Finally, hit the Compile and Build buttons on the menu bar.

Having compiled the code, you can run it (with the green running person icon), then load the tracefile back into the simulator and watch it run (use the clock icon, and choose the relevant results file). Run it to watch changes happen in the simulated construction. You can also watch the pipelining happen, and the virtual pages being requested and loaded.

If you want to look at the program instructions themselves, they are found in the **DRUM\_STORE.pageX.mem** files in the **model** directory, starting with page 0. They are structured as:

#### instruction Ba Bm address

The Drum Store contains the program code in page 0, fixed-point integers in page 2, and floating-point reals in page 3. The Core Store is empty at the start of the simulation, with Block 0 modelled as an instruction array, Block 1 as an integer array, and Block 2 as a floating-point array. As each array of code/integers/floating-point number is needed, it is fetched in from the Drum Store.

For an explanation of the instructions used in each model, check out the HASE Atlas simulation webpage ([www.icsa.inf.ed.ac.uk/cgi-bin/hase/atlas.pl?menu.html,atlas.html](http://www.icsa.inf.ed.ac.uk/cgi-bin/hase/atlas.pl?menu.html,atlas.html)), which also has more details of the simulation itself. The listing of the first demonstration program doubles as a list of Atlas instructions.

You can also try the other demonstration programs, both of which do actual mathematics. V3.2 is a Sum of Squares program, which should report in the Output window the result 3, 4, 5, (then print **stopping**). This is the solution of the equation **a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup> for a, b, c < 8**. We couldn't find any output for the Matrix Multiplication program (updates would be welcome if any readers do experiment with it!). An explanation of the model is at the link above.

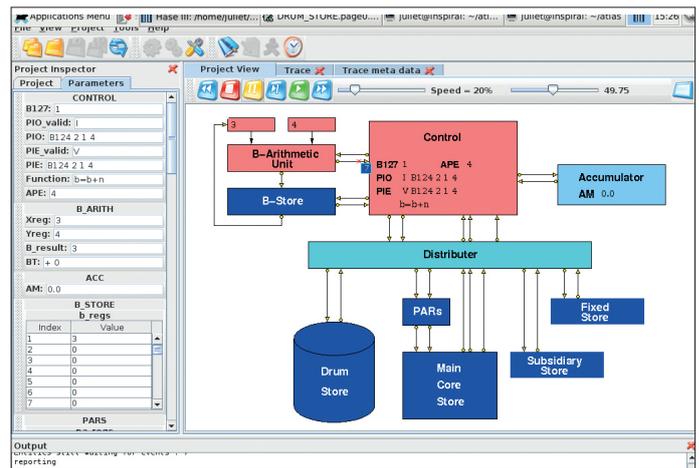
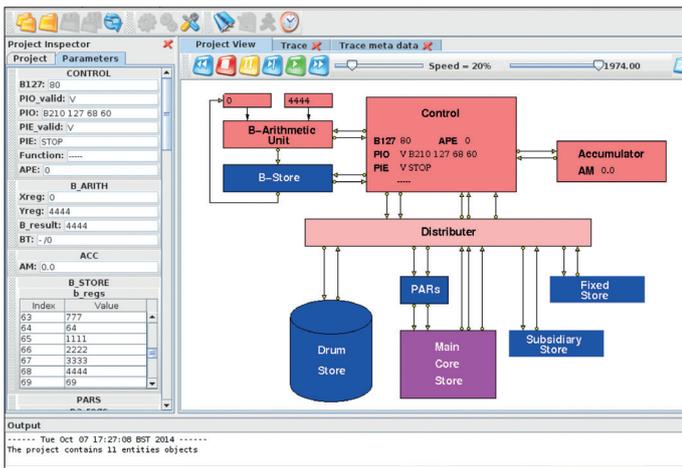
More information about HASE, a user guide, and how to create your own models, is available here: [www.icsa.inf.ed.ac.uk/research/groups/hase/manuals/index.html](http://www.icsa.inf.ed.ac.uk/research/groups/hase/manuals/index.html).

### Building your own program

If you copy the contents of one of the sample directories wholesale into another directory, and rename **atlas\_v\*. \* to my\_project.\*** (so you're renaming the **.edl .elf .params** files), you can edit the **DRUM\_STORE.page0.mem** file to produce your own small program. Here's one example:

```
A314 0 0 12288
A320 0 0 12296
A346 0 0 12304
STOP
nop 0 0 0 ... to end of file (must be 256 lines)
```

This loads the value in word 1536 into the accumulator (**A314**), adds the value in word 1537 to it (**A320**), and then stores the result in word 1538 (**A346**). Words 1536 onwards are found at the start of



**DRUM\_STORE.page3.mem**, and are loaded into the core store block 2 when needed.

To run it, load the project, compile it, run, and then you can watch the trace. If editing, reload the project, then recompile.

Here's the same example (adding two numbers) in B-instructions:

**B121 1 0 3**

**B124 2 1 4**

**E1064 0 0 0**

**E1067 2 0 0**

**STOP**

This loads the value 3 (not a memory address) into B1 in line 0 (**B121**), then in line 1 adds 4 modified by the contents of B1 to B2:

**B124 B-register B-modification-register Number**

So in practice this adds  $4 + B1 = 4 + 3 = 7$  to B2 (which starts as zero). Line 2 uses an Extracode instruction:

**E1064**

to output a newline, then line 3 uses another Extracode instruction:

**E1067**

to output the contents of B2.

You can see the output 7 in the bottom window, and in the main window, the 7 in the process of being returned to Control as part of instruction 1.

As mentioned above, the first test program listing in the Atlas model information ([www.icsa.inf.ed.ac.uk/cgi-bin/hase/atlas.pl?menu.html,atlas.html](http://www.icsa.inf.ed.ac.uk/cgi-bin/hase/atlas.pl?menu.html,atlas.html)) is effectively an instruction listing. The earlier B instructions, for example, access memory locations rather than absolute values. Remember that A instructions managed floating point operations, and B instructions the integer operations. This means that A instructions operate only on the floating-point values (from block 2 of the core store, word 1536 onwards, memory location 12288 onwards), and B instructions only on integers (block 1 of the core store, word 1024 onwards, location 8192 onwards). We're not sure to what extent this exactly mirrors the real setup of Atlas memory and to what extent it is a feature of the organisation of the simulator, but do bear it in mind to avoid getting really frustrated with memory locations

that won't load! Two more Atlas machines were built alongside the Manchester one; one shared by BP and the University of London, and one for the Atlas Computer Laboratory in Chilton near Oxford, which provided a shared research computing service to British scientists.

## After Atlas

Ferranti also built a similar system, called Titan (aka Atlas 2), for Cambridge University. Its memory was organised a little differently, and it ran a different OS written by the Computer Lab folk at Cambridge. Titans were also delivered to the CAD Centre in Cambridge, and to the Atomic Weapons Establishment at Aldermaston. The Manchester Atlas was decommissioned in 1971, and the last of the other two closed down in 1974. The Chilton Atlas main console was rediscovered earlier this year and is now at the Rutherford Appleton Laboratory in Chilton; National Museums Scotland in Edinburgh also has a couple of its parts. The Titans closed down between 1973 and 1974.

The Atlas team were responsible for the start of numerous concepts (such as pipelining, virtual memory and paging, as well as some of the OS ideas behind Atlas Supervisor) which are still important in modern computing; and, of course, at the time, the machines themselves were of huge research importance. It's rather a shame that it seems largely to have been forgotten in the shadow of other supercomputers such as those made by Cray and by IBM. It was certainly a very successful British project at the time.

In 2012, Google produced a short film remembering the Atlas, which is available online. There's also a collection of links and memories available on the Manchester University website. There's some documentation on the Chiltern Computing website, too, including this brochure from 1967 ([www.chilton-computing.org.uk/acl/literature/acl/p003.htm](http://www.chilton-computing.org.uk/acl/literature/acl/p003.htm)).

Juliet Kemp is a scary polymath, and is the author of *O'Reilly's Linux System Administration Recipes*.

The simulator after running the v1.3 model. The blue fast-forward button runs the whole thing as fast as possible. The green button allows you to watch more slowly, or you can step through one process at a time.

## LV PRO TIP

There is an emulator (which copies external behaviour) of the whole thing available, but it's Windows-only; see Dik Leatherdale's webpage at [www.dikleatherdale.webspace.virginmedia.com/atlas.html](http://www.dikleatherdale.webspace.virginmedia.com/atlas.html).

# PYTHON: MAKE YOUR OWN JULIA FRACTALS

With a smattering of Python and some clever maths you can create even more beautiful chaos inside your Linux box.

Julia sets are closely related to the Mandelbrot Set, which we explored last issue with some basic Python. All of these are produced by the code we'll develop here from the same simple iterated function  $z^2+c$  as the Mandelbrot fractals.

You can see that they are different to the Mandelbrot set, and yet there's something about them that is similar. You can also see that some Julia sets are a single object like the Mandelbrot, but some are many pieces, with some even becoming so fragmented they are almost like dust.

It might not be obvious, but while there is only one Mandelbrot, there are infinitely many Julia sets. The recipe, or algorithm, for creating Julia sets is the same as that for the Mandelbrot set except for one key difference.

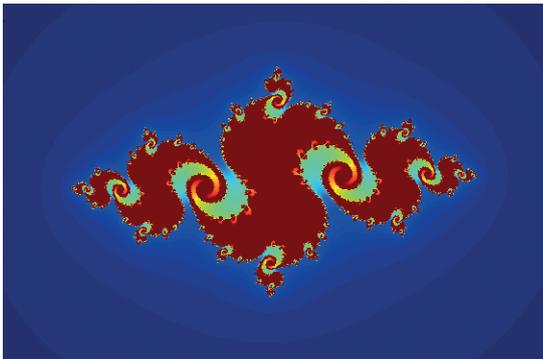
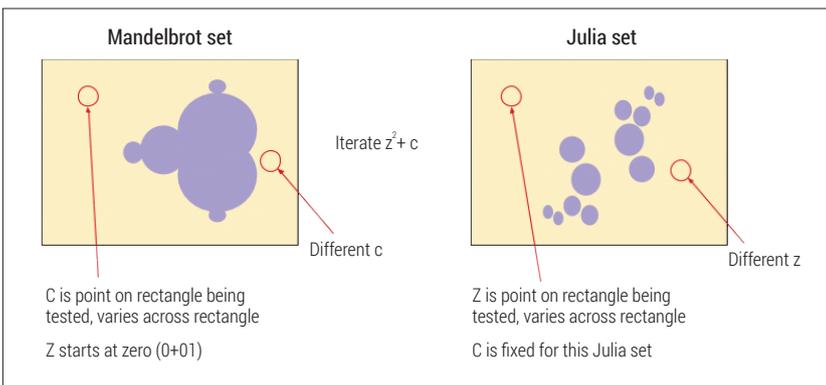
Let's remind ourselves of the recipe for the Mandelbrot set. We choose a rectangle on the complex plane and test many evenly spaced points to see whether they're part of the Mandelbrot set. We do this by seeing if repeated iteration of a function  $z^2+c$ , where  $c$  is the point being tested, results in the orbit escaping and diverging, or not. And if it does, we colour the point according to how fast the point diverges.

This recipe is essentially the same for making Julia sets. The difference is that the roles of the  $z$  and  $c$  values are reversed (see diagram below).

The following shows the calculating function `julia(z, c, maxiter)`, based on the `mandel(c, maxiter)` function we developed in the last issue:

```
def julia(z, c, maxiter):
 for iteration in xrange(maxiter):
 z = (z*z) + c
 if abs(z) > 4:
 break
```

Instead of  $c$  representing the point on the complex being tested (as in the Mandelbrot set), it is kept constant for all the points being tested. Similarly instead of  $z$  starting from zero, it starts as the complex number representing the point being tested.



Just one look tells you that the Julia set is very closely related, but subtly different to the Mandelbrot set.

```
pass
pass
return iteration
You can see that z doesn't start at (0+0i) anymore but is a parameter passed to the function. The parameter c is also passed but is a fixed complex number and not a number representing the point being tested, because z does that now.
The Python program to plot Julia sets is presented here for easy reference:
loads the numerical and plotting extensions to Python
needed if you're using a locally installed IPython, not for some online IPython providers
%pylab inline
set the location and size of the atlas rectangle
xvalues = linspace(-2, 2, 1000)
yvalues = linspace(-2, 2, 1000)
size of these lists of x and y values
xlen = len(xvalues)
ylen = len(yvalues)
value of c (unique for each Julia set)
c = complex(-0.35, 0.65)
julia function, takes the fixed parameters z and c and the maximum number of iterations maxiter, as inputs
def julia(z, c, maxiter):
 # start iterating and stop when it's done maxiter times
 for iteration in xrange(maxiter):
the main function which generates the output value of z from the input values using the formula (z^2) + c
```

```

z = (z*z) + c

check if the (pythagorean) magnitude of the output
complex number z is bigger than 4, and if so stop iterating as
we've diverged already
if abs(z) > 4:
 break
 pass
 pass

return the number of iterations we actually did, not the final
value of z, as this tells us how quickly the values diverged past
the magnitude threshold of 4
return iteration

create an array of the right size to represent the atlas, we use
the number of items in xvalues and yvalues
atlas = empty((xlen,ylen))

go through each point in this atlas array and test to see how
many iterations are needed to diverge (or reach the maximum
iterations when not diverging)
for ix in xrange(xlen):
 for iy in xrange(ylen):

 # at this point in the array, work out what the actual real and
 imaginary parts of x are by looking it up in the xvalue and yvalue
 lists
 zx = xvalues[ix]
 zy = yvalues[iy]
 z = complex(zx, zy)

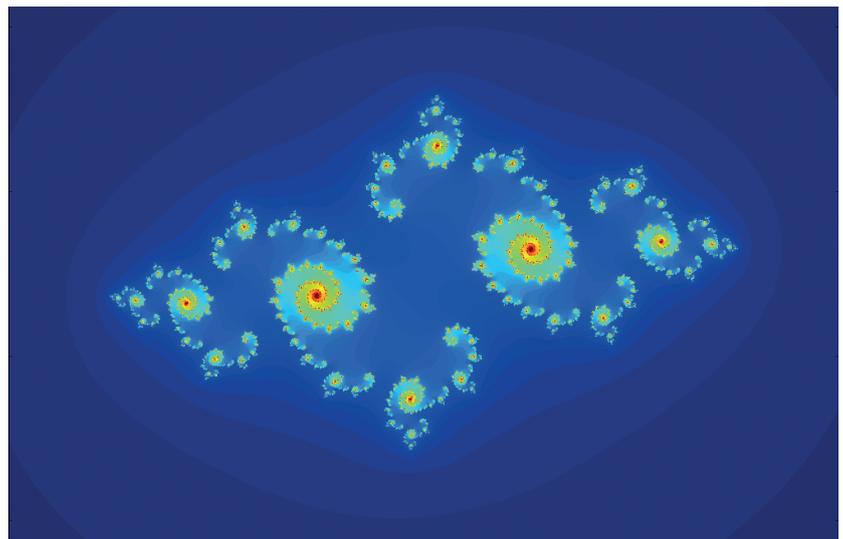
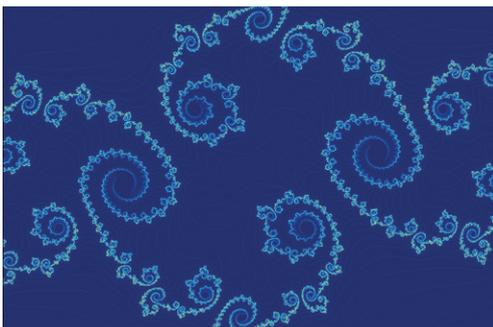
 # now we know what c is for this place in the atlas, apply the
 mandel() function to return the number of iterations it took to
 diverge
 # we use 80 maximum iterations to stop and accept the
 function didn't diverge
 atlas[ix,iy] = julia(z,c,80)

```

## Pretty as a picture

Here's an image produced by one of the filters provided by the same extension that we got the Gaussian filter from. It's beautiful – and is the cover of the author's ebook (<http://www.amazon.co.uk/Make-Your-Mandelbrot-Tariq-Rashid-ebook/dp/B00JFIEC2A>). See if you can recreate it yourself.

The image is the result of a filter that highlights edges (a Sobel filter) applied to the same Julia images we created previously. The code is at <http://bit.ly/1qpnl5E>.



Some Julia fractals are more broken up, while others are more contiguous.

```

 pass
 pass

set the figure size
figsize(18,18)

plot the array atlas as an image, with its values represented as
colours, peculiarity of python that we have to transpose the
array

imshow(atlas.T, interpolation="nearest")

```

Try plotting different Julia sets by changing the **c** parameter. Remember it is the single parameter **c** which uniquely defines the Julia set.

## 3D fractal landscapes

This section will require you to use a locally installed version of *IPython*, because the online *IPython* services can't access your OpenGL graphics subsystem to render the three-dimensional plots.

Let's try to turn the two-dimensional flat images of the Mandelbrot and Julia sets into three dimensions. One way to do this is to use the colour information in each image array to represent altitude; the height of a landscape above sea level as it were. We should then see mountainous landscapes shaped by the values of arrays filled by the Mandelbrot or Julia calculations.

## Surface plot

Let's try it on a very simple array first. The following code prepares a 3x2 array:

```

a = zeros([3,2])
a[0,0] = 1
a[0,1] = 2
a[1,0] = 4
a[2,1] = 3
print a

```

Plotting a simple flat image based on these values, using `imshow(a, interpolation="nearest")` results in the simple 2D plot as expected.

### PRO TIP

The code for the full programs to calculate and plot the Mandelbrot and Julia fractals is online at <http://bit.ly/1qpnl5E>.

Now let's plot this same array in three dimensions, with the third dimension given by the value of each cell. To do this we need to use a new extension to Python called **mayavi**.

Telling *IPython* that we want to use **mayavi** uses the **import** instruction, just as before. The special **pylab** instruction is to ensure that the common set of numerical extensions are automatically loaded in the local *IPython*:

```
%pylab inline
import mayavi.mlab

The instructions to plot a surface are simple:
mayavi.mlab.surf(a, warp_scale="auto")
mayavi.mlab.show()
```

The first instruction prepares the surface plot and allows *IPython* to automatically scale the heights. The second instruction is needed to actually show the plot. This time the result will appear in a separate window outside your browser. You can do quite a lot to this 3D plot, including rotating it around and changing the lighting that illuminates it. The images at the bottom of the page show the above small array plotted in this way, and again rotated using the mouse. Try it yourself.

For easy reference, the entire code for creating a simple 3x2 array and plotting a three-dimensional plot of it as follows:

```
%pylab inline
import mayavi.mlab

a = zeros([3,2])
a[0,0] = 1
a[0,1] = 2
a[1,0] = 4
a[2,1] = 3

mayavi.mlab.surf(a, warp_scale="auto")
mayavi.mlab.show()
```

We now apply this same idea to the larger arrays created by the Mandelbrot code we wrote earlier. We simply add the new instructions to create the

three-dimensional plots, remembering to import the **mayavi** extension too. It's worth keeping the previous **imshow()** instruction to see the flat view too. The instructions are as follows. The **warp\_scale** was adjusted to 1.0 to give a reasonable height, the default setting created hills that were a little too steep:

```
mayavi.mlab.surf(atlas.T, warp_scale=1.0)
mayavi.mlab.show()
```

These three-dimensional height fields as some people call them, give an interesting perspective of the Mandelbrot set.

Let's try that again but with the original rectangle zooming into details of the Mandelbrot set, and also change the colour palette by using the interactive menus. Some of these landscapes are quite haunting. Let's try the same idea with Julia sets.

### Gentler landscapes

Some of the landscapes we created were a little spiky and noisy. Let's see if we can calm them down, smooth off the sharp edges, and reduce the cragginess a little, all in the hope of creating a perhaps more gentle, more appealing, landscape.

One way to do this is to calculate a new image array, based on the original one, but with each new value somehow smoothed based on its neighbour's values. We want to calculate the average over a small group of neighbouring array elements, perhaps a 3x3 square or a bigger circle. Engineers and scientists who work with images or signals do this kind of thing fairly often to try to reduce noise.

Python provides a collection of such filters that can be applied to an array of values, which is often useful when they are in fact images. A common smoothing filter is a slightly more sophisticated version of the local average we just described, called a Gaussian blur filter. If you work with photo editing applications like *Photoshop* or *Gimp* then you'll be familiar with applying a Gaussian blur to smooth an image.

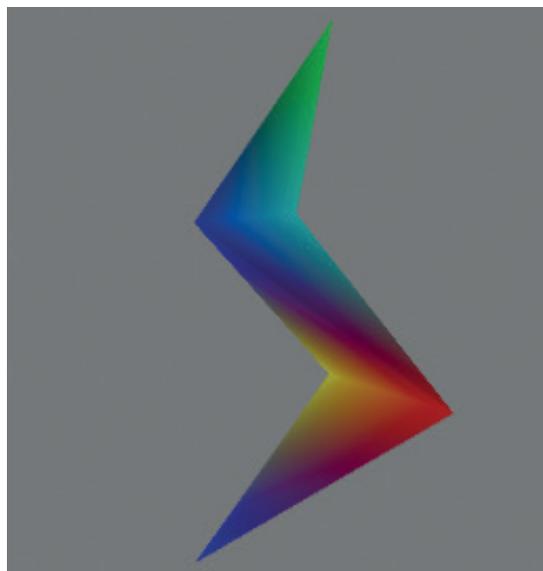
To get access to these filters, we'll need to import the Python extension at the top of our code as follows:

#### LV PRO TIP

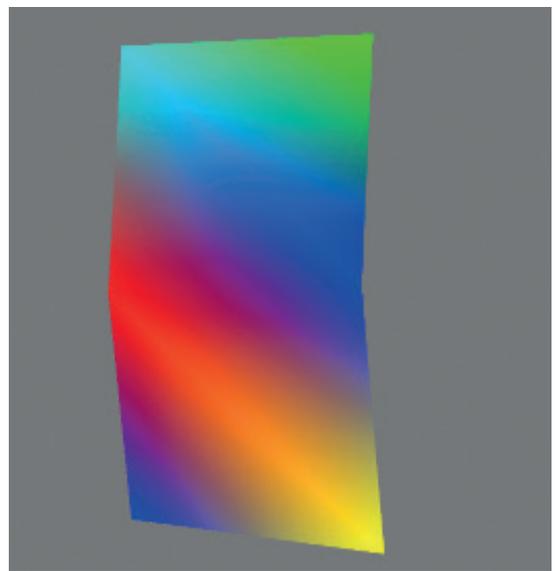
Don't forget that in Python, like many programming languages, counting elements of arrays and lists starts at 0 not 1.

#### LV PRO TIP

You can explore these 3D views by rotating them and dragging them with your pointer, and use the menus to select different lighting, colour palettes and many other options.



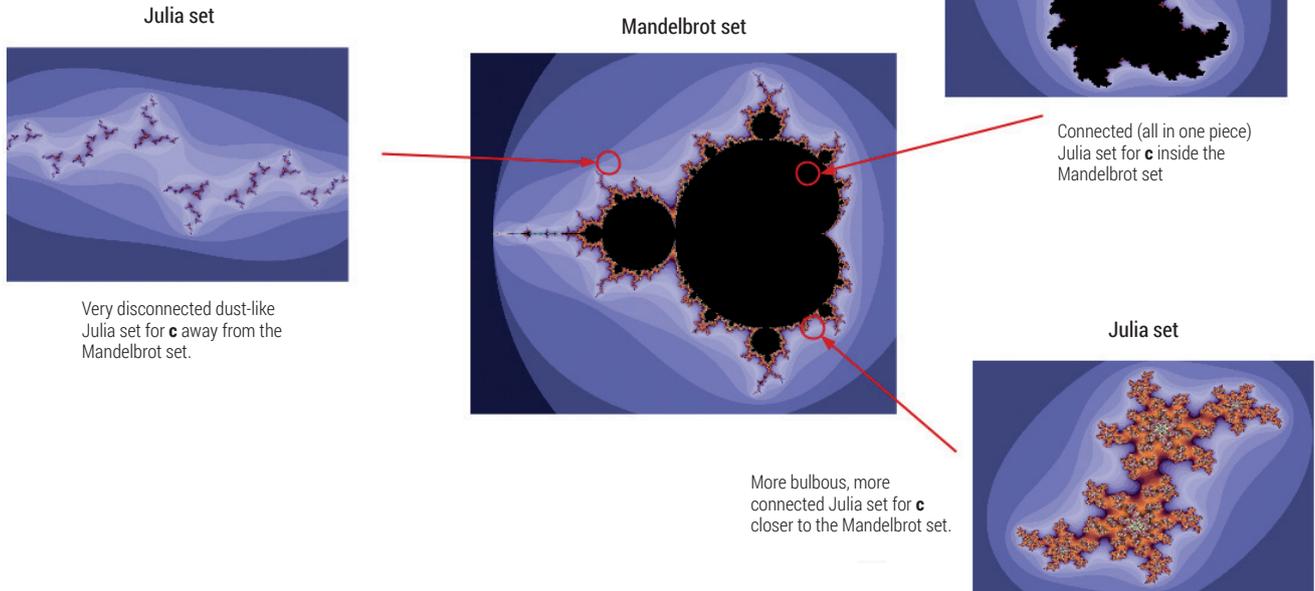
Small rotated and illuminated 3D plotted array.



## The intimate relationship between Mandelbrot and Julia fractals

The Mandelbrot and Julia fractals are intimately connected. The relationship is illustrated in the following diagram. Each Julia fractal is uniquely defined by the chosen value of  $c$  in  $z^2+c$ . For values of  $c$  that fall inside the Mandelbrot fractal, the Julia

fractals are a single object. For values of  $c$  that fall outside the Mandelbrot, the Julia fractals are many separate pieces. The further  $c$  moves from the Mandelbrot set, the more fragmented and dust-like the Julia fractals become.



### import scipy.ndimage

The following shows how the Gaussian blur filter can be used to create a smoothed image array from the original, which we called atlas in our previous code. The value 2 is the strength of the smoothing:

```
smoothed_atlas = scipy.ndimage.gaussian_filter(atlas.T, 2)
```

We can plot this as a flat image just as before using the **imshow** function:

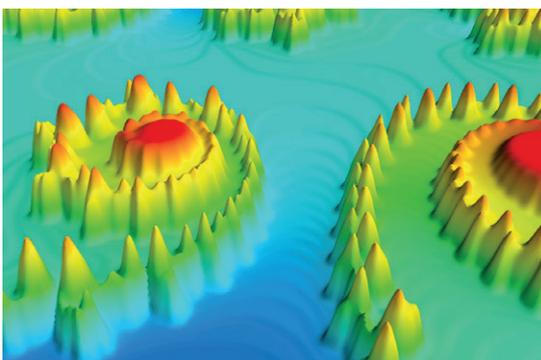
```
set the figure size
```

```
figsize(18,18)
```

```
create a smoothed image of the original by applying a Gaussian blur filter
```

```
smoothed_atlas = scipy.ndimage.gaussian_filter(atlas.T, 2)
```

```
plot the array atlas as an image, with its values represented as colours, peculiarity of python that we have to transpose the array
```



Add a 3D height field and a Gaussian blur to our Mandelbrot image and you get the Misty Mountains.

### imshow(smoothed\_atlas, interpolation="nearest")

Similarly we can plot the 3-dimensional landscape just as before using the **mayavi** extensions.

```
mayavi.mlab.surf(smoothed_atlas, warp_scale=0.9)
```

```
mayavi.mlab.show()
```

We'd suggest playing with the settings for the Gaussian smoothing filter to get an effect that works for you. For example, we've had great results applying the filter to the Julia set generated by  $c=(-0.768662 + 0.130477i)$ , but there are many different variations and many kinds of smoothing effects you can produce.

### Try It Yourself

We'd encourage you to explore other filters to see if they make interesting images. Perhaps you might explore different functions aside from  $z^2+c$  which was the basis for both the Mandelbrot and Julia fractals.

And remember, you can't do any harm by experimenting and playing. We hope in this series of tutorials you've discovered the fun and convenient *IPython* environment for web-based Python programming, and become familiar with the basic building blocks of Python programming. We also hope you've experienced some of the surprise, excitement and beauty that even simple mathematics holds, particularly if your experience has been of boring trigonometry puzzles. Now go and play with some maths! 

### PRO TIP

You can explore more filters, besides the Gaussian blur, to apply to your images at <http://bit.ly/1mNyMWE>

Tariq spends his time grappling with enterprise IT problems, informed by two decades of working with open technology.

# CODE NINJA: STDIN AND STDOUT

Make your Python programs work with the Linux shell to build your own command line utilities.

## WHY DO THIS?

- Understand one of the basic principles of how the command line works.
- Write your own utilities for the Linux shell.
- Get more out of Python.

If you've used the Linux command line for more than the most basic uses, you'll be familiar with the concepts of standard in (**stdin**) and standard out (**stdout**). These are channels the shell pushes data through when you link commands with the pipe character (**|**). For example :

**dmesg | grep "USB"**

Here, the first command (**dmesg**) sends the kernel message buffer to **stdout**. The **|** symbol takes the **stdout** of one command and pushes it into the **stdin** of the next. The **grep** command then goes through each line in its **stdin** and prints out only those containing the string 'USB'. Using this concept of **stdin** and **stdout**, we've managed to build a simple little tool for checking the kernel messages about USB devices.

In the Unix philosophy, as much as possible, programs should be simple commands that work with **stdin** and **stdout**. If they do, then they can be combined with the other Unix commands in a huge number of ways. Working this way means you can

focus your program on doing one thing well. Rather than having to program every possible feature, you just focus on the core of the program and the user can use other Unix tools to gain

more features by chaining commands together.

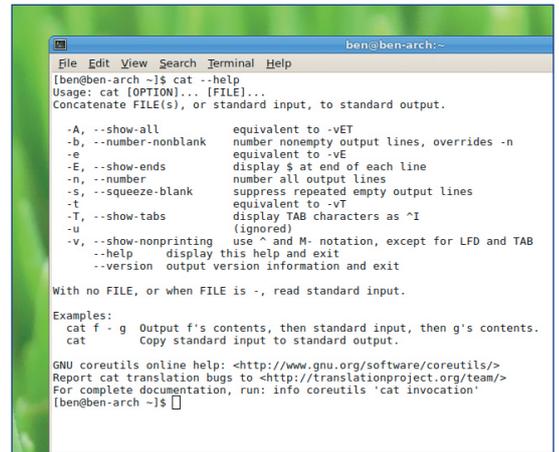
Let's take a look at how we could start building our own implementations of standard Unix tools in Python that interact using **stdin** and **stdout** so they can be chained together.

Let's start with **cat**. This stands for catenate. Basically, it's used to send the output of one or more files to **stdout**. In Python, this is really easy:

```
from __future__ import print_function
import sys

for arg in sys.argv[1:]:
 with open(arg) as file:
 for line in file:
 print(line, end="")
```

All we have to do to send something to **stdout** in Python is print it. Here we've used the **print** function, which is the standard way of doing it in Python 3. This also works in Python 2 if you include the first import line. The reason we've done it this way is because we'll use some features of the **print** function a little later on, and want to keep things consistent.



GNU's **cat** utility has many more features than we've implemented here, but it still works on the same basis of reading files and sending their contents to **stdout**.

Each line that we read from the file includes the newline character at the end, so we include the parameter **end=""** to stop the **print** function from adding a separate newline character which would leave a blank line between every line from the file.

## In and out

We can get the arguments to the command from **sys.argv**. This is simply a list of everything that's supplied to the Linux Shell (see "Arguments" box). Here, we just use this to loop through every argument to the command and hope it's a file. So, the following would send the output of **file1** and **file2** to **stdout**:

**python cat.py file1 file2**

You can chain this with other Unix commands such as **wc** (word count) like this:

**python cat.py /usr/share/dict/words | wc**

**wc** outputs the number of lines, words and characters in a file. Let's implement that next:

```
from __future__ import print_function
import sys

num_lines = 0
num_words = 0
num_chars = 0

for line in sys.stdin:
 num_lines += 1
 num_words += len(line.split())
 num_chars += len(line)
```

## Arguments

Interacting nicely with the Linux command line environment isn't all about using **stdin**, **stdout** and **stderr**. Another aspect of the command line environment is properly parsing arguments and flags. For example, in the **cat** Python script, we take a list of files as arguments. We did this using **sys.argv**. This provides a list of everything that's passed to the command (the first item in the list is the command itself; in that example, we only included everything from the second element onwards).

However, if your command takes a range of options, it can be quite complex to correctly parse this. Fortunately, there's a module to help: **argparse**. Let's take a look at a simple example. This will be a command a little like **echo**, but it will send the output to **stdout** unless the **--stderr** flag is present, in which case it will send it to **stderr** (this is just an example, and the normal **echo** command doesn't have these flags).

We'll do this using the **argparser** module (new in Python 2.7, so this won't work if you're using an older version). Using this, you only have to specify what options you have, and the module will take care of parsing them and putting them in variables so you can access them:

```
from __future__ import print_function
import argparse, sys
parser = argparse.ArgumentParser(description='echoing to stdout
and stdin')
```

```
parser.add_argument('echo', help='The stuff to print on screen')
parser.add_argument("--stderr", help="print the stuff to stderr
(otherwise print to stdout)", action="store_true")
args = parser.parse_args()
if args.stderr:
 print(args.echo, file=sys.stderr)
else:
 print(args.echo)
```

You can now check it's working with the following lines:

```
python echo.py --stderr "hello world" | wc
python echo.py "hello world" | wc
```

The **argparser** module even takes care of creating your help text (from the description and help parameters used when creating the parser), so you can run:

```
$ python echo.py --help
usage: echo.py [-h] [--stderr] echo
echoing to stdout and stdin
```

positional arguments:

```
echo The stuff to print on screen
```

optional arguments:

```
-h, --help show this help message and exit
--stderr print the stuff to stderr (otherwise print to stdout)
```

You can fine tune **argparser's** operation to almost any degree, and it has excellent documentation at: <https://docs.python.org/2/howto/argparse.html>.

```
print("\t", num_lines, "\t", num_words, "\t", num_chars)
```

As you can see, getting input from **stdin** is quite straightforward. It's read in in lines just like files are.

These two tools can be linked together with:

```
python cat.py /usr/share/dict/words | python wc.py
```

So far, we haven't implemented any error checking. So, if you try to **cat** a file that doesn't exist, you get a Python exception:

```
$python cat.py /nosuchfile | python wc.py
```

This might seem a little strange, because the error from the first Python command hasn't been sent to **stdin** of the second command. Instead it's been displayed on the screen. The reason for this is that errors don't go to **stdout**: instead, there's a separate channel for them called standard error (**stderr**). This means that a program can send error messages

without corrupting the information going to **stdout**.

For example, you might have used **cat.py** to output the contents of several files, but one of them doesn't exist. Using **stderr**, you still send the output of all the files to the next command in the chain, but also alert the user to the problem. Let's take a look at how to modify our **cat** command to deal with this:

```
from __future__ import print_function
import sys, os.path
for arg in sys.argv[1:]:
 if not os.path.isfile(arg):
 print("whoops, there appears to be a mistake. I can't find",
arg, file=sys.stderr)
 else:
 with open(arg) as file:
 for line in file:
 print(line, end="")
whoops, there appears to be a mistake. I can't find sdxhfsd
from __future__ import print_function
import sys
num_lines = 0
num_words = 0
num_chars = 0
for line in sys.stdin:
 num_lines += 1
 num_words += len(line.split())
 num_chars += len(line)
print("\t", num_lines, "\t", num_words, "\t", num_chars)
```

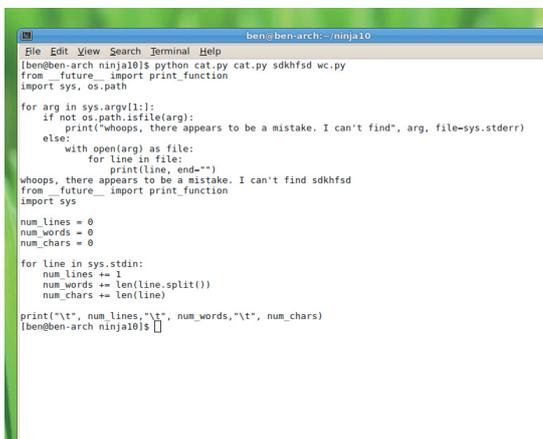
As you can see, all you have to do to print something to **stderr** is include the parameter **file=sys.stderr** in the **print** function, and you can print to it just like you print to **stdout**.

**Stderr** doesn't have to be printed on the screen though; you can send it to a file instead. If you're running commands through some automated means such as **at** or **cron**, it can be useful to collect any errors. This is done with:

```
command 2> file
```

Anything a command sends to **stdout**, will be printed on the screen (you could also pipe it into further commands). Another option is to tell **Bash** to send **stderr** to **stdout**. Doing this will combine **stdout** and **stderr** into a single flow of text and pass it along to any future commands:

```
command1 2>&1 | command 2
```



Even if one of the arguments to our **cat** program isn't a file, it will still handle the rest correctly.

# ROBOCODE: ARTIFICIAL INTELLIGENCE WARFARE

A computer game that's played by computers? Is this an open invitation to Skynet or the future (or both)?

## WHY DO THIS?

- Satisfy your power cravings by programming a robot army.
- Learn Java in a well documented environment.
- Give yourself a chance to win a Linux Voice T-shirt.

The *Robocode* IDE is fine for simple robots, but if you're creating a more complicated warrior, you may prefer to use a more fully-featured IDE.

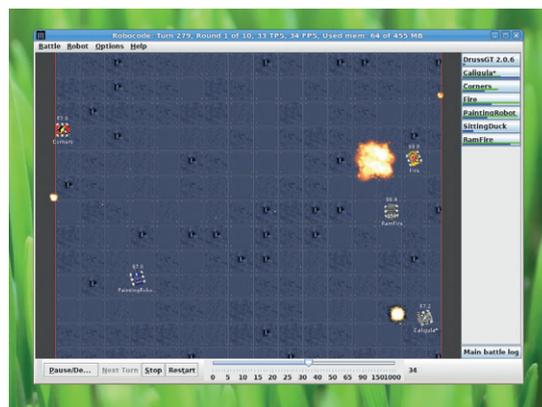
The idea of creating robotic warriors is as old as science fiction. However, how hard would they be to make? Let's not worry about the physical things – that's a problem for hardware engineers – and take a look at the software side of things. *Robocode* is an environment for tank-based autonomous warfare. That means that rather than controlling the actions of your tank via a joystick or keyboard during the fight as you may in a regular computer game, you have to program it to behave in the way you want it to, then set it running. Before we look at how to program the tanks, let's first take a look at the *Robocode* environment.

The main website is <http://robocode.sourceforge.net>, where you'll find loads of useful information that will come in handy if you decide to take on the challenge. There's also a download link that will take you to a SourceForge page where you can grab the JAR file. To use this, you'll need to have Java installed (we used OpenJDK 7, but other versions may also work).

Once it's downloaded, run:

```
java -jar robocode-1.9.2.3-setup.jar
```

This will pull down all the files you need, and create a directory for them to live in. Once it's finished, you



Let battle commence! Watching battles at normal speed helps you understand how the battles are fought, but cranking up the speed lets you test your robots quickly.

can **cd** into this directory (by default it will be **~/robocode**), and start the software with:

```
./robocode.sh
```

*Robocode* comes with a range of sample robots, so the first thing is to run a few battles to see how it works, and what tactics are effective. Go to Battle > New to create a new battle. On the first tab, pick a few robots to fight it out (it doesn't really matter which ones; we find that about five is a good number of competitors). The second tab allows you to change the settings for the battle (number of rounds, size of battlefield, etc). You can leave these as the defaults. Press Start Battle to begin.

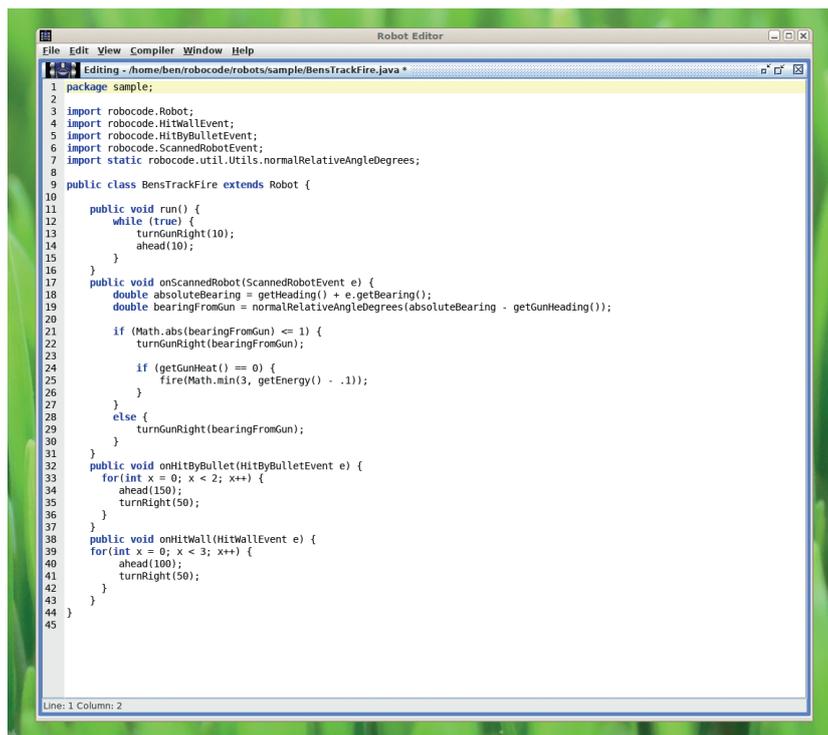
## Know your robots

You should see that each robot uses different tactics to attack the others. Some (like *RamFire*) are really aggressive, while others (like *Corners*) are more defensive. Essentially, there are two parts to a robot's strategy. It needs to avoid the shells from other tanks, and it needs to hit other tanks with its shells.

Before we dive in and start writing our own robot, let's take a look at how a couple of others work. *Robocode* has its own built-in IDE, which you can use to create robots in Java. Got to Robot > Source Editor to open it, then Open > Sample > *RamFire.java* to take a look at how the *RamFire* robot works. It is (in slightly abridged form):

```
public class RamFire extends Robot {
 int turnDirection = 1;

 public void run() {
```



```

while (true) {
 turnRight(5 * turnDirection);
}

public void onScannedRobot(ScannedRobotEvent e) {
 if (e.getBearing() >= 0) {
 turnDirection = 1;
 } else {
 turnDirection = -1;
 turnRight(e.getBearing());
 ahead(e.getDistance() + 5);
 scan();
 }
}

public void onHitRobot(HitRobotEvent e) {
 if (e.getBearing() >= 0) {
 turnDirection = 1;
 } else {
 turnDirection = -1;
 }
 turnRight(e.getBearing());
 fire(1);
 ahead(40);
}
}

```

There are quite a few comments that we've removed for brevity. The first thing that surprised us was just how simple the code is. *Robocode* handles everything about the environment, so you only need to decide how your robot will react to the environment and code the AI.

Standard robots extend the **Robot** class (there are other classes such as **AdvancedRobot** you can use as well), and then override the appropriate methods. In the case of **RamFire**, there are only three methods.

**Run** is triggered at the start of the battle and is used to define the tank's default behaviour. In this case, if nothing else happens to the robot, it will simply spin around on the spot.

**OnScannedRobot** is triggered when the robot's scanner picks up another robot. Control of the radar (which picks up other tanks when scanning) can be a complex topic, but in simple terms, it will just look in the general area the turret is facing. Because the tank will continually spin until directed otherwise, it will always be on the look out for other tanks. This method simply points the tank at the scanned tank, and head straight for them.

When the robot rams into its victim, **onHitRobot()** will run. This fires the cannon at them (the code here is simplified), and it will ram them again.

As you can see, controlling your robot is all about understanding the *Robocode* API. This is fully documented at <http://robocode.sourceforge.net/docs/robocode>. You don't need to dive straight in and read it all, but as you develop fighting robots, you'll probably find yourself heading there more and more frequently.

**RamFire** works by aggressively pursuing robots – a tactic that can work, but can also incur damage for

## Advanced tactics

Robots vary in complexity from the simple (like the ones we're looking at) to the mind-boggling complex. Fortunately, you don't have to start out from scratch when building your robot. There's lots of information on tried and tested tactics on the *Robocode* wiki ([http://robowiki.net/wiki/Main\\_Page](http://robowiki.net/wiki/Main_Page)). Here are a few of our favourite tactics:

- **Energy drop tracking** Robots can't see when bullets are fired, but they can detect the amount of energy other robots have. If that energy drops suddenly, it's usually because the other robot has fired a bullet. This can be useful to other tactics such as wave surfing and bullet shielding.
- **Wave surfing** If you see a robot fire a bullet

(by energy drop tracking), you don't know what direction it's gone in, but you can predict a range of directions, and work out the rough probability of each. These areas of probability ripple outwards like waves from a stone dropped in a lake. Wave surfing is the process of getting in the place of the lowest probability on this wave. There's more details at: [http://robowiki.net/wiki/Wave\\_Surfing\\_Tutorial](http://robowiki.net/wiki/Wave_Surfing_Tutorial)

- **Bullet Shielding** The aim of bullet shielding is to protect your robot by shooting your opponents' bullets out of the air. That's quite easy to say, but much harder to achieve in practice. See [http://robowiki.net/wiki/Bullet\\_Shielding](http://robowiki.net/wiki/Bullet_Shielding) for the maths.

the **RamFire** robot itself. Now let's take a look at a more cautious robot. **TrackFire** also continuously looks for enemies, but rather than ramming them, it just shoots at them:

```

public class TrackFire extends Robot {
 public void run() {
 while (true) {
 turnGunRight(10);
 }
 }
 public void onScannedRobot(ScannedRobotEvent e) {
 double absoluteBearing = getHeading() + e.getBearing();
 double bearingFromGun = normalRelativeAngleDegrees(absoluteBearing - getGunHeading());
 }
}

```

The screenshot shows the LiteRumble website in a Mozilla Firefox browser. The page displays a table of participants for various categories:

| 1v1                         |                        | Participants |
|-----------------------------|------------------------|--------------|
| <a href="#">roborumble</a>  | <a href="#">top 50</a> | 1088         |
| <a href="#">minirumble</a>  | <a href="#">top 50</a> | 596          |
| <a href="#">microrumble</a> | <a href="#">top 50</a> | 461          |
| <a href="#">nanorumble</a>  | <a href="#">top 50</a> | 272          |
| <a href="#">gigarumble</a>  | <a href="#">top 50</a> | 30           |

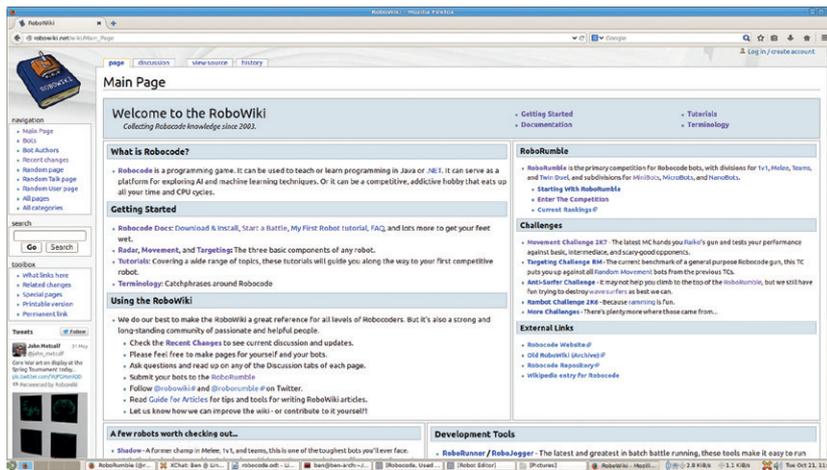
| Melee                            |                        | Participants |
|----------------------------------|------------------------|--------------|
| <a href="#">meleerumble</a>      | <a href="#">top 50</a> | 385          |
| <a href="#">minimeleerumble</a>  | <a href="#">top 50</a> | 187          |
| <a href="#">micromeleerumble</a> | <a href="#">top 50</a> | 144          |
| <a href="#">nanomeleerumble</a>  | <a href="#">top 50</a> | 83           |
| <a href="#">meleeTop30rubble</a> | <a href="#">top 50</a> | 30           |

| Teams                      |                        | Participants |
|----------------------------|------------------------|--------------|
| <a href="#">teamrubble</a> | <a href="#">top 50</a> | 44           |
| <a href="#">twinduel</a>   | <a href="#">top 50</a> | 26           |

Below the tables, there are links for [LiteRumble Statistics](#) and [Score Explanation](#). At the bottom, a note states: "The LiteRumble costs ~\$2/week in server time and database access. If you want help with these costs, please donate via Paypal." with a [Donate](#) button.

There's an ongoing *Robocode* competition at <http://litterumble.appspot.com>. If you think your creation is up to the challenge, you can set it against the rest of the world and see how it matches up.



The robot wiki is an invaluable source of information and inspiration for all things *Robocode*. [http://robowiki.net/wiki/Main\\_Page](http://robowiki.net/wiki/Main_Page)

```

if (Math.abs(bearingFromGun) <= 3) {
 turnGunRight(bearingFromGun);
 if (getGunHeat() == 0) {
 fire(Math.min(3 - Math.abs(bearingFromGun),
getEnergy() - .1));
 }
}
else {
 turnGunRight(bearingFromGun);
}

if (bearingFromGun == 0) {
 scan();
}
}
}

```

As you can see, this is quite simple. Rather than spinning the robot, it just turns the gun, and rather than speeding off towards its victim, it just points the cannon at them and fires.

There is a little complication with firing because you (as the programmer) can decide how much power to give the gun. The more power (the parameter to the **fire** command), the more damage it potentially does your enemy, but the more it also heats up your gun. If your gun overheats, you can't fire until it cools down.

A good general tactic is to fire with more power the more confident you are of hitting your enemy. In this example, we'll fire if the angle between where the gun is pointing, and the tank we're trying to hit is less than three degrees. However, the smaller the angle is, the more power we'll give the shot. This isn't necessarily optimal because it doesn't attempt to work out whether the target is moving.

If you watch TrackFire fight a few battles, you'll probably notice that the biggest problem it has is that it can be a sitting duck. Once it turns up in an enemy's scan, it is usually destroyed fairly quickly because it doesn't get away.

Now you've seen a few robots in operation, and seen how a couple are written, it's time to start

building one. We decided to base ours on the TrackFire robot, but give it a bit more motion and the ability to run away.

To start with, then, we just need to update the parts that identify the robot, so open the TrackFire robot in the source editor if you haven't already, and change the **public class** line to:

```
public class LVTrackFire extends Robot {
```

Then go to File > Save As and save it under a new name (it should suggest **LVTrackFire.java**, which is fine). You now have a new robot. To make sure everything's gone properly, go to Compiler > Compile, then (if there are no errors), go to the main *Robocode* window and create a new battle. You should find your robot under Sample.

## Added intelligence

Of course, at this stage, the AI for LVTrackFire is exactly the same as TrackFire, so let's now go back and add some more intelligence to it.

The biggest problem TrackFire has is that it stays still. We'll add a bit of movement logic to our robot in a few ways:

- Under normal operation, it will move forward slowly. This will help it avoid fire from robots far away.
- If it's hit, it will quickly get out of the way. This will hopefully move it out of the scan of the robot that's just hit it.
- If it hits a wall, it will get away. This stops it being pinned in by RamFire.

We can add this logic in stages. Firstly, we'll make it move forward under normal operation. To do this, just add one line to the **run** method as follows:

```

public void run() {
 while (true) {
 turnGunRight(10);
 ahead(10);
 }
}

```

## Famous fighters

The robots that come with *Robocode* are enough to keep you interested while you get started, but if you really want to test your skills, you may want to test yourself against some more powerful opposition. Alternatively, you may just want to watch some masterful killing machines at work. Whichever is your motive, you can find more robots online at <http://robowiki.net/wiki/Category:Bots>. A few of our favourites are:

- **DrussGT** ([http://minify.rchomepage.com/robocode/jk.mega.DrussGT\\_2.0.6.jar](http://minify.rchomepage.com/robocode/jk.mega.DrussGT_2.0.6.jar))
- **Diamond** ([www.dijitari.com/void/robocode/voidious.Diamond\\_1.7.11.jar](http://www.dijitari.com/void/robocode/voidious.Diamond_1.7.11.jar))
- **Demonic Rage** ([http://sites.google.com/site/justinsitehere/file-cabinet/justin.DemonicRage\\_3.4.jar](http://sites.google.com/site/justinsitehere/file-cabinet/justin.DemonicRage_3.4.jar))

If you want to see some top *Robocode* action, put Demonic Rage in a ring with DrussGT and Diamond to see which comes out top. If you can build a robot that can compete with these three, you've done very, very well.

These are all open source, so you can poke around in their insides and see what makes them tick. Perhaps you'll find something to inspire your own battle bot.

}

This keeps the gun spinning as before (and so keeps it scanning for other robots), but makes it move as well.

The next bit of code we want is to make the robot run away whenever it's hit. Again, we don't really care where it runs away to, we just want it get away from its current location. In this case, we could just move forwards a certain distance. This is what the first iteration of our robot did, but in a crowded battlefield, we found that this doesn't work very well. Our tank often got stuck when it hit other tanks and didn't run away successfully, so instead, we made it move forwards, turn, then move forwards again. This is done with the following method:

```
public void onHitByBullet(HitByBulletEvent e) {
 for(int x = 0; x < 2; x++) {
 ahead(150);
 turnRight(50);
 }
}
```

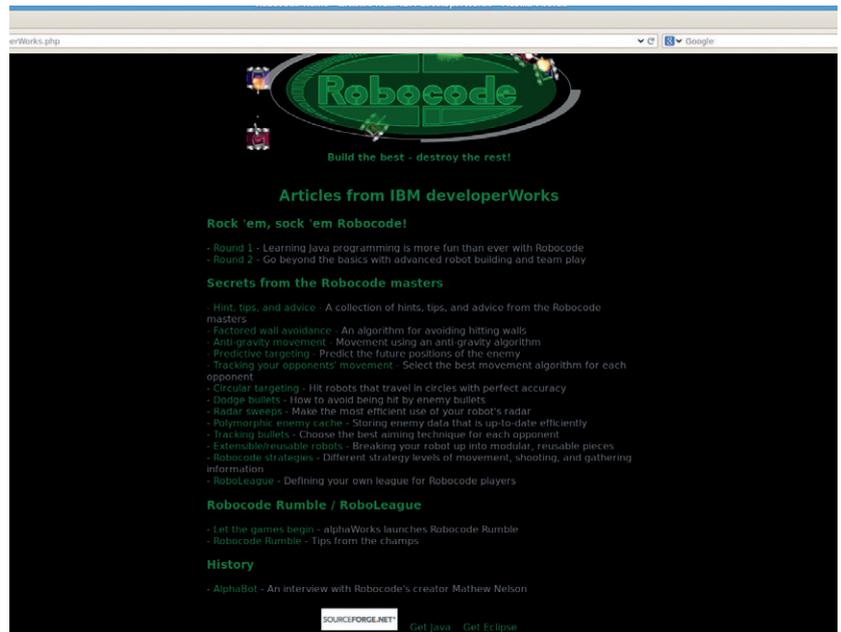
The final bit of movement (to keep the robot away from walls) is done with:

```
public void onHitWall(HitWallEvent e) {
 for(int x = 0; x < 3; x++) {
 ahead(100);
 turnRight(50);
 }
}
```

This is the mechanics robot complete. We were happy with TrackFire's firing, so we didn't change this. You can compile it and pit it against other robots in battle and see how it fares.

Although it's all done, there are still some bits you can tweak to make the robot more effective.

The speed at which our robot moves is governed by the number in `ahead(10)` in the `run` method. By increasing or decreasing this number, you can make it faster or slower. You can also adjust the accuracy of the aiming in the line:



```
if (Math.abs(bearingFromGun) <= 3) {
```

Reducing the value `3` will mean the gun will wait until it has pointed more accurately before it fires. This will reduce the rate of fire, but increase the number of bullets that reach the target (since the target could be moving, it's not easy to aim accurately).

You can also adjust the firepower. None of these change the fundamental operation of the robot, but by tuning the parameters, you can make your killing machine more effective.

Our simple robot isn't the most devastating opponent, but hopefully it will have whetted your appetite to build your own AI tank. It should also prove a fairly easy starting point to surpass as you add more features, you should quickly be able to superseded this primitive AI. 🤖

**Ben Everard is the best-selling author of *Learning Python With Raspberry Pi*. He hacks robots for fun.**

IBM's Developer Works has a series on learning Java with *Robocode* at <http://robocode.sourceforge.net/developerWorks.php>. It's a little dated now, but should still work.

## COMPETITION

We're going to run a digital Battle Royale.

There are lots of open source robots available, so we're going to make our challenge a bit different to most *Robocode* competitions to discourage the use of these open source bots.

Firstly, your tank must extend the `Robot` class, not `AdvancedRobot` or any of the others available in *Robocode*. Second, the whole robot must come in at under 70 lines of Java. A line (for the purposes of this challenge) is a statement ended by a semicolon (not including the separate parts of a `for` loop), or a statement that starts a new code block (such as `if`, `else`, `while` etc). Parentheses may be put at the beginning or end of any line and don't need a separate line. Therefore, the shortest the following code can be is three lines:

```
for(int x = 0; x < 3; x++) {
 ahead(100);
 turnRight(50);}
```

Obviously, the best tactics will depend on how the battles are structured. In our war, the battle will take place in a series of heats. Each heat will have six robots (if the number of robots isn't divisible by six, sample robots will be included to bulk up the numbers). Each heat will be 100 rounds, and the top three will progress to the next round. We'll go through as many rounds as necessary to whittle the field down to six robots for the grand final. This will take place over 200 rounds and the winner of this will be crowned Linux Voice Robocode champion. The winner will, of course, get an exclusive Linux

Voice winner's T-shirt. All the battles will take place on a 1000x800 battlefield with a gun cooling rate of 0.1, an inactivity time of 450 and a sentry size of 450.

All robots must be licensed under a OSI-approved open source licence (we recommend the GPL v3). The robot doesn't have to be entirely your own work: borrowing chunks of code from other openly licensed robots is allowed, as is using existing robots as a starting point for your development, although you obviously must adhere to the terms of the licence of any code you borrow.

Please email your robot's source code to [ben@linuxvoice.com](mailto:ben@linuxvoice.com) by 20 December 2014.



# MASTERCLASS

Without music, life would be a mistake. Join us as we celebrate the new UK legal status of ripping your own CDs.

## DVD-VIDEO: RIP AND BURN...

Now that you can legally copy DVDs, we show you how.

JOHN LANE

On 1 October, the UK's copyright law caught up with the real world. You can now legally make personal copies of your favourite ebooks, music and videos that you own. So, without further ado, this month's Masterclass will have you making backups of your favourite DVDs in no time. We'll explain the structure of a DVD and show you how to make your own so that you can now make that movie you've always wanted to.

There are two ways to copy a DVD. The first is to create an ISO image of the whole disc – an exact copy. Most media players will play images just like the real thing and you can burn them onto blank media to make an exact copy. The other way is to extract the contents of the DVD's filesystem by recursively copying its root directory. Again, most media players will play a directory tree copied from a DVD.

Before you start copying, make sure you have sufficient free storage. A typical DVD weighs in at around 7GB – you can see the actual size of a disc with the **isozise** command:

```
$ isozise /dev/sr0
6654584832
```

This outputs the raw size of the disc's filesystem in bytes (our optical drive is **/dev/sr0**).

If you just want to make an ISO image, you may be able to do it with the **dd** or **cp** commands. This is the

### Install the packages

Some of the tools that we will use are included in the base packages of most distros (for example, things like **dd** and **isozise**, which we use to make ISO images). You'll need some other tools to go beyond, however.

- **lsdvd** displays the contents of a DVD; it's a quick way to identify the main title on a disc, because this is usually the longest track.
- **vobcopy** copies the contents of a DVD as files and directories.
- **libdvcss** decrypts titles that use the Content Scramble System (CSS).
- **ffmpeg** is used to transcode video files into the format required by DVD-Video.
- **cdrtools** provides the tools we need to write ISO images to disc.



The UK Government now communicates changes in the law of the land via the proprietary medium of Twitter. If it is Tweeted, it must be official.

most basic method – it copies the raw data from the disc's block device into an image file.

The following commands are equivalent, as they copy the entire disc:

```
$ dd if=/dev/sr0 of=mydvd.iso
```

```
$ cp /dev/sr0 mydvd.iso
```

It's possible for these methods to receive a few null blocks at the end of the stream. They won't affect the image but, for an exact block-perfect copy, you can try these instead:

```
$ dd if=/dev/sr0 of=mydvd.iso bs=2048 count=$(isozise -d 2048 /dev/sr0)
```

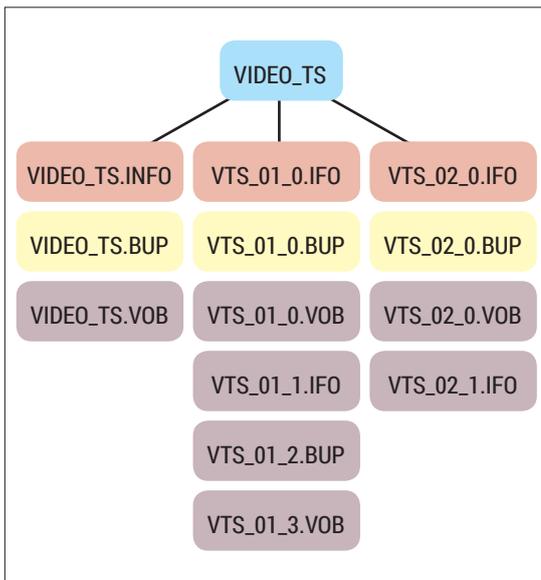
```
$ head -c $(isozise /dev/sr0) </dev/sr0 >mydvd.iso
```

These use **isozise** to obtain the exact size of the original and then copy that amount of space (the block size of a DVD is 2048 bytes). A **md5sum** hash of the original disc and images produced this way should match.

### Authoring

If you encounter errors then you might be able to overcome them by adding **conv=noerror,sync** to the end of the **dd** command. These arguments make the **dd** command continue after errors (**noerror**) and NUL-pad the error blocks (**sync**) to preserve the image size. Both these methods of using the **dd** command provide a disc image suitable for playback on many players, such as **VLC**; they provide a true backup of the whole disc, complete with any encryption and copy protection they may have.

If you want to modify your copy, perhaps by extracting only the main title to save space or by



Each title on a DVD has files collectively called a Title Set; there can be up to 99 title sets on one disc. This example has two; the first is divided into four VOB files and the second one is divided into two.

transcoding for playback on other devices, then you'll need to read the data files from the disk and work with those instead. The long-standing tool of choice for this is the **vobcopy** command, which requires that you mount the DVD first:

```
$ sudo mount /dev/sr0 /mnt
```

```
$ vobcopy --mirror /mnt
```

The DVD will be extracted to a new directory created inside the current directory, whose name will be determined by the titling on the DVD (usually the name of the programme on the disc). If the files are encrypted and the appropriate libraries are installed then **vobcopy** will decrypt them first so that it can copy them and the resulting copies will be free from encryption.

## A look inside

If you look at an extracted DVD-Video disc you will see that there is a directory named **VIDEO\_TS** at its root (there may also be an **AUDIO\_TS** directory, which is for DVD-Audio discs and is normally empty on a DVD-Video disc).

The **VIDEO\_TS** directory contains three kinds of files: IFO (information) files tell the player how to navigate the disc and have BUP backups (**x.ifo** and

## Intentionally bad sectors

The DVD copier's nemesis is the intentional bad sector. These are sectors that have incorrect CRC checksums, and they are used as a crude copy protection on some commercially produced discs. Most standalone players don't encounter the errors because they are avoided by the disc indexes (the IFO files) they follow but they are hit by attempts to image the disc.

There are several such schemes, with names like ARccOS, XProtect and RipGuard. If you encounter such a disc, you should be able to make an ISO from it with the GNU **ddrescue** tool, like this:

```
$ ddrescue -n -b 2048 /dev/sr0 mydvd.iso
```

The **-n** parameter causes bad blocks to be

ignored; they're intentional and contain no data so we don't care about them. The block size of a DVD is 2048 bytes.

If the image is still unplayable, extract it with **vobcopy** or **dvdbackup** and play that instead. If you still want an ISO, perhaps to write to another disc, you could use **mkisofs** to make one from the extracted directories.

An alternative method is to locate the main title and stream it into a file:

```
$ lsdvd /dev/sr0
```

```
longest track is 4
```

```
$ mplayer dvdnav://4 -dvd-device //dev/sr0
```

```
-dumpstream -dumpfile main_title.mpg
```

**ddrescue** may also help you rip a scratched disc.

**x.bup** are identical). The third kind of file is the VOB (Video Object), and these contain the actual media: video, audio and subtitles. They're basically MPEG-2 Program Stream files with DVD-specific limitations. Any player that can play MPEG can play unmodified VOB files. Because they can't be larger than 1GB, longer titles are split into multiple files. You can, however, use **cat** to join them:

```
$ cat mydvd/VIDEO_TS/VTS_01_*.VOB > title.mpg
```

The files follow a strict naming convention: the **VIDEO\_TS** files are for the Video Manager and contain the control and playback information for the disc, referring to title sets which are the **VTS\_nn\_n** files on the disc. Once you understand the format, you can make your own DVDs by creating the appropriate files.

## Authoring

Taking regular video files and making a DVD from them is called authoring. Your source material may be in various formats and will probably need to be transcoded first; one suitable tool is **ffmpeg**:

```
$ ffmpeg -i my_video.mp4 -target pal-dvd my_dvd_video.mpg
```

The simplest authoring just takes your video file and packages it into a DVD-Video directory structure suitable for the PAL video system used in the UK:

```
$ export VIDEO_FORMAT=PAL
```

```
$ dvdauthor -d my_dvd -t my_dvd_video.mpg
```

```
$ dvdauthor -d my_dvd -T
```

The last command writes the DVD table of contents and completes a basic example that will play a single title when the disc is inserted. You can create an XML configuration file to define chapter marks and multiple titles or add menu systems - read the documentation at <http://dvdauthor.sourceforge.net> to learn more.

Before you can give that holiday video to your gran, you need to make an ISO image and put it onto a disc:

```
$ mkisofs -dvd-video -o my_dvd.iso my_dvd
```

```
$ cdrecord -dao dev=/dev/sr0 my_dvd.iso
```

```
$ cdrecord -fix dev=/dev/sr0
```

Fixating the disc is the final step and makes it ready for use in standalone players. If it doesn't happen automatically the last command will do it.

### LV PRO TIP

You can use **cp** on its own to copy unencrypted files from a mounted DVD.

### LV PRO TIP

**dvd+rw-mediainfo /dev/sr0** shows useful information about the disc.

## tccat: another way to extract a title

You can use **cat** to extract a DVD title, but the resulting MPEG header will have incorrect duration data, which can cause confusion in some players. The **transcode** package includes a utility called **tccat** that offers another way to achieve a similar result direct from a disc or image:

```
$ tccat -i /dev/sr0 -T1,-1 > output.mpg
```

The **-T** option specifies the track to extract. The format is **-T<track>,<chapters>** and specifying a chapter value of **-1** selects the whole track.

# DVD-VIDEO: RIP AND BURN

The GUI way to read and write your video DVDs.

JOHN LANE

You are somewhat spoilt for choice if you prefer to use a desktop application to read and write DVD-Video media. If you're looking for ease of use the current favourite is *Handbrake*, an open-source, GPL-licensed but Apple-flavoured video transcoder. If all you want to do is extract content from a DVD to watch on your phone, tablet, desktop or smart television then look no further than your distro's repository.

*Handbrake* comes in command line and *GTK* desktop versions; the executable for the latter is called **ghb**. When you start it, the main screen is displayed and is where you select source material, desired encodings and other settings.

You can choose from almost any video file format supported by **libavformat** and **libavcodec**. The output file formats are limited to MKV and MP4, but these can contain various video encodings including H.264, MPEG-4 and MPEG-2 along with the usual audio encodings that go with them. For use with most modern devices, it probably has your needs covered.

To make it even easier for you, the right-hand side of the main window displays presets, mostly for Apple and Android phones and tablets, but there is also the default preset (Normal) and a general-purpose preset for H.264 video (High Profile), with all the bells and whistles. You can also add, modify and remove presets, whether they're built-in or your own.

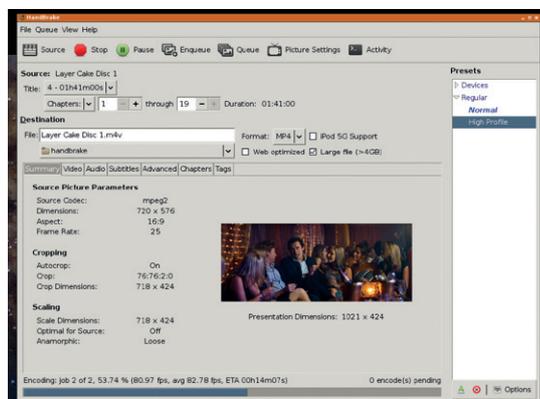
Copying a DVD is easy: place it in your drive and press the Source button to select it (this is also how you'd choose material on your hard drive). Once selected, the disc is scanned and the title list is populated. *Handbrake* attempts to select the main title for you, and you can use the Title drop-down menu to select any title. You can select part of the title by chapter numbers, time in seconds or frames. Choose your desired preset, set the destination and press

**LV PRO TIP**

*Handbrake* writes MP4 files with a **.m4v** extension so *QuickTime* and *Qt 4* applications can play them properly. There is no internal difference.

**LV PRO TIP**

Advanced users can tweak *Handbrake's* x264 encoder parameters; they are documented at <https://trac.handbrake.fr/wiki/x264options>.

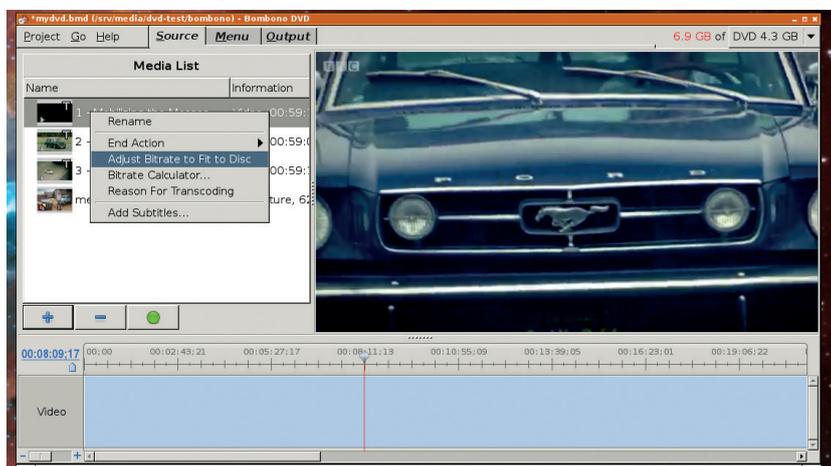


Handbrake at work: the Start button changes to Stop while it's working. It takes around 30 minutes to transcode a feature film.

Start. That's all there is to it. Because *Handbrake* copies and also transcodes, or converts from one format to another, it takes longer than a simple copy would, but it works and gets the job done without a great deal of fuss.

## Make your own DVD

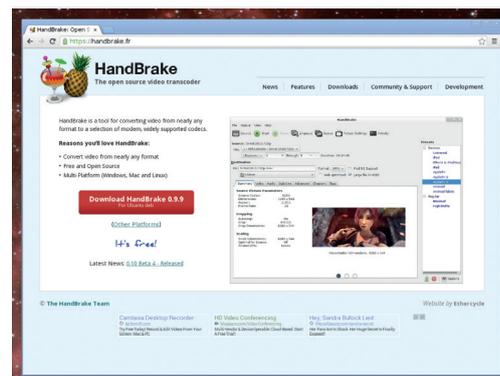
One thing that *Handbrake* doesn't do is DVD authoring – this is the preparation of video in the required format and creating the menu system used to navigate the disc. There are quite a few tools that purport to offer these capabilities. One of them is called *Bombono DVD* and should be in your distro's repository. Its *GTK* user interface is straightforward, having three main tabs that are used to select the source material, build a menu system and, finally, write the disc. The writing stage can include writing



Bombono's video timeline lets you move through a title and set chapter marks.

## Support options

Handbrake benefits from decent documentation and support resources. There's a user guide, forums and you can head over to **#handbrake** on Freenode IRC and <https://handbrake.fr>.



the DVD directory tree, creating the disc image (ISO file) and burning the image to a disc.

The first stage is where you choose your content; you don't need to worry about its format because it will be transcoded into the required MPEG-2 Program Stream format that DVD-Video needs. You also don't need to worry about its size, because you can adjust its bitrate to fit on the destination media. To add media, you can either drag it on to the left-hand Media List panel or use its Add Media Files button to select them from a pop-up list. You can re-order selected titles by dragging and renaming them (although this has nothing to do with the DVD-specific names used on the disc).

You can right-click on a title to choose what should happen when it ends; the default action is to return to the menu. You also right-click and use 'Adjust Bitrate To Fit To Disc' to make sure your content will fit.

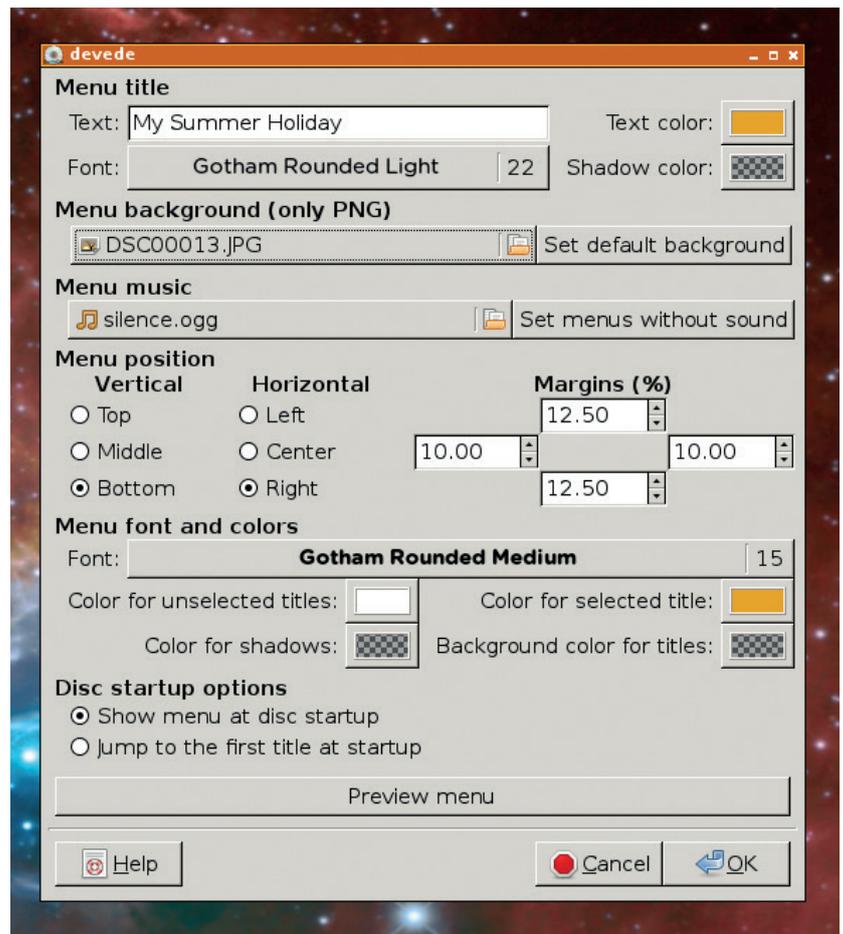
## Menus

The next stage is to prepare menus. You can set a background image and write text labels or drag and drop chapters to produce a set of thumbnails that play from those points in the title, and there are basic align and distribute tools to help you lay them out neatly. Menu creation is something you could quite easily get carried away with, but can yield good results to give your disc a polished look and feel.

Writing is the final stage and uses the command-line **dvdauthor** utility underneath. You have to select a working directory and it's a good idea to create a new, empty, one for this. You can choose whether to only create the DVD folder – the directories and files that will go on the disc – to create an image or burn onto a disc. If your content needs to be transcoded, the process will take longer.

Another popular tool that performs similar tasks is *DeVeDe*: you select your content, create menus and launch it. A little while later, your choice of a disc structure or ISO pops out. There's no option to burn that to a disc, so you'll need to use something else for that task.

The menu options are more limited than *Bombono's*: you can only have one menu and the layout options are much more primitive. It displays each title as text



You can create menus with *DeVeDe* but the options are limited – there are no dragging elements here and no submenus, but it is quick and easy.

(no thumbnails here) and you can choose where they appear: at the top, middle or bottom and either on the left, right or in the centre. You can also place an overall title across the top of the menu screen and select a soundtrack and a background image (this needs to be prepared to the correct aspect ratio beforehand or will appear distorted). A preview window shows you what your menu will look like.

Once your disc image is written, you can preview it in a player such as *VLC* to make sure it's OK before using a separate burning tool to write it to a disc.

## Spoilt for choice; spoiled by cruft

If you search your repositories for graphical GUI tools, you'll find options of varying quality, many of which are either abandonware that will present problems as soon as you try to install them, or feature-poor, knocked-together efforts that fail to meet their objectives. This is one area of open source software where there is a significant amount of cruft.

That said, the ones that we've used here work well and do what they were designed to do. Have faith – the package manager will provide.

**John Lane provides technical solutions to business problems. He has yet to find something that Linux can't solve.**

### LV PRO TIP

*Handbrake's* preset parameters are described at <https://trac.handbrake.fr/wiki/BuiltInPresets>.

## A few alternatives

- **MakeMKV** is dedicated to ripping DVD-Video to Matroska MKV containers.
- **PGCEdit** is a DVD IFO and menu editor designed to allow the modification of the navigation commands and parameters of an already authored DVD structure.
- **AcidRip**, a *GTK* wrapper for the **mencoder** command, is feature-rich but one for the more advanced user.
- **Brasero** is a beginner-level tool that is a standard part of the *Gnome* desktop.
- **K3b** is the *KDE* application for ripping and burning media. And then there's *VLC*, which can do just about anything. But that flexibility cost is met by complexity that may be too much for the casual user taking a backup of the occasional DVD.

# LINUX VOICE DVD 010

Don't fall behind – get the latest Linux goodness today!



## SAY HELLO TO THE UTOPIC UNICORN

Ubuntu has had its ups and downs over the years, but it remains one of the most popular distros of all, and is the entry point into Linux for many people. It's polished, it's built on a solid foundation (Debian) and there are huge support communities on the internet in case you have problems. If you've picked up this magazine for the first time and you're thinking about

installing Linux, we can highly recommend Ubuntu.

But it's not just for newbies: I personally run Xubuntu as my main production distro. It has a great out-of-the-box experience, a super fast and slick desktop, good hardware detection and huge package repositories.

On another note, as this is Linux Voice issue 10, we're giving away

our very first issue under the Creative Commons (CC-BY-SA). This means you're free to share it and create new content based on it – just give us credit for making it in the first place! See [index.html](#) on the DVD for more information. Happy Linuxing!

Mike Saunders, Disc Editor  
[mike@linuxvoice.com](mailto:mike@linuxvoice.com)

### Distro powerpack

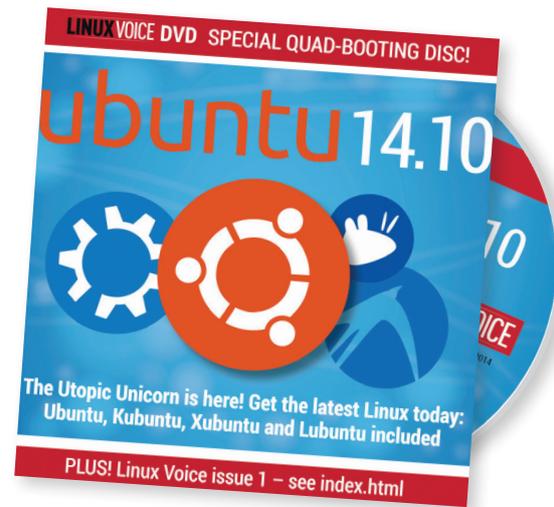
## Ubuntu 14.10

Full 64-bit version, directly bootable from the disc

Yes, it's that time of year again: another Ubuntu release is here. This one isn't particularly radical in terms of changes, but that's fine with us; it's not necessary to have a giant upheaval every six months. Ubuntu 14.10 is an incremental and evolutionary release, with a handful of new features backed up by package updates, bugfixes and refinements. If you're running a previous release, it's worth the upgrade just to get the latest packages – or if you're a newbie in the Linux world, it's a great way to get started. You can find out more about the latest release in our review on page 48, so here we'll focus on installation.

The version of Ubuntu 14.10 on our DVD is 64-bit, and you'll need at least 1.5GB of RAM and 10GB of hard drive space. If you've never installed Linux before and you only have Windows on your drive, you can choose to shrink the Windows chunk of the drive (its partition) during the installation, to make room for Linux. Then you'll be able to choose your operating system when you boot your PC. Before installing, though, it's a good idea to back up important data!

If you're looking for something with lower system requirements, or you want to squeeze extra performance out of your PC, try one of the other Ubuntu variants



mentioned below. And if you get stuck at all, consult the friendly Ubuntu community online at [www.askubuntu.com](http://www.askubuntu.com).

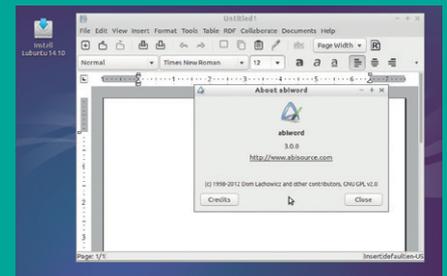
## Xubuntu, Kubuntu, Lubuntu

And there's more! Ubuntu re-spins galore

This month's DVD is a quad-booter, so along with the regular Ubuntu version, it can also boot Kubuntu, Xubuntu and Lubuntu. Just choose the one you want from the Grub menu when you boot the disc, and you'll soon arrive at your choice of desktop.

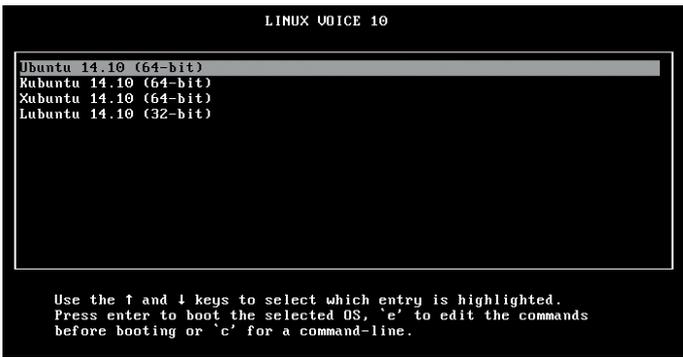
Kubuntu (64-bit) uses the KDE desktop instead of Unity – this is a very mature

environment, generally considered the most flexible and featureful in the Free Software world. Xubuntu (also 64-bit) uses the faster and lighter Xfce desktop. And for older machines, we've added Lubuntu (32-bit) which only needs 512MB of RAM, and is bundled with various memory-friendly desktop applications.

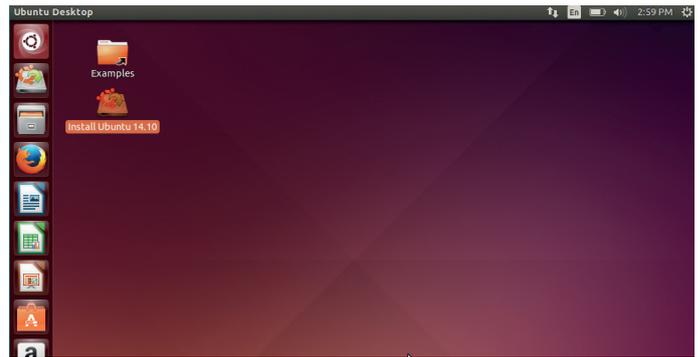


Lubuntu ([www.lubuntu.net](http://www.lubuntu.net)) is great for breathing new life into old PCs.

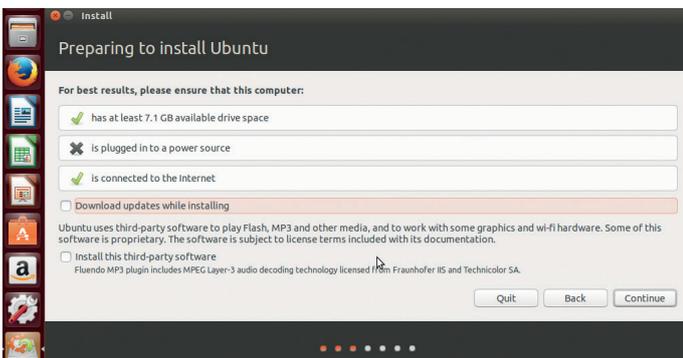
# How it works: Installing Ubuntu 14.10 from the DVD



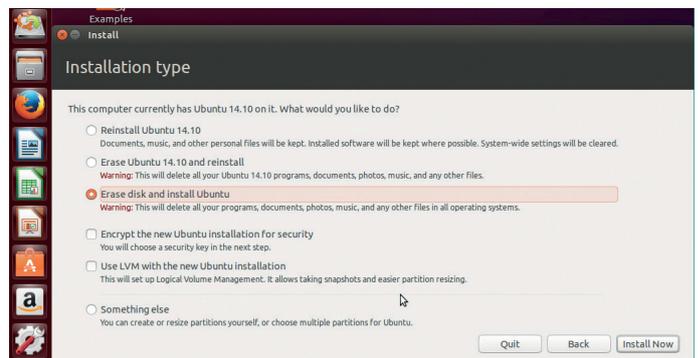
**1** Boot your PC from the DVD, and you should see this menu. (If not, you may need to change the boot order in your BIOS – see your PC's manual.)



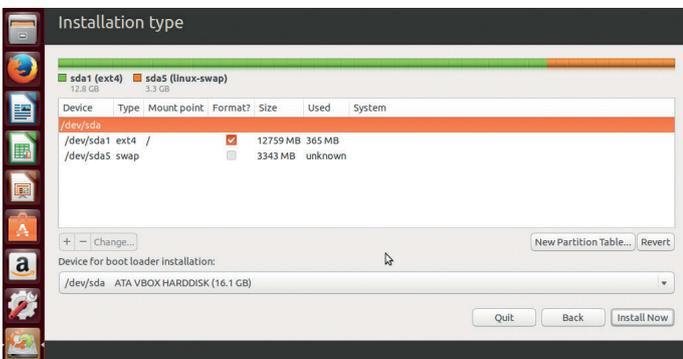
**2** After choosing the Ubuntu flavour you want, you'll arrive at the desktop. This is live mode, so you can explore the software before installing.



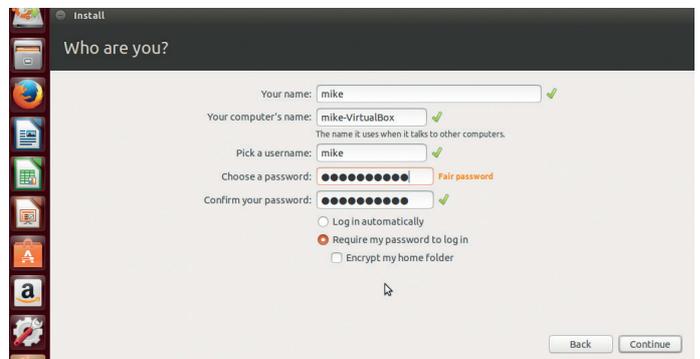
**3** Double-click the installer icon on the desktop and the installer will pop up and ask if you want to download updates and third-party add-ons.



**4** Next, partition (divide up) your hard drive: dedicate it all to Ubuntu, share it with another Linux distro or Windows or something else (custom setups).



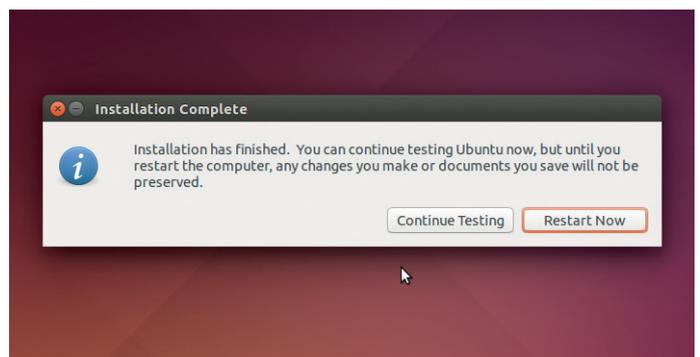
**5** Advanced users: if you do custom partitioning, you'll need at least a root (/) partition along with swap (twice your RAM, but no bigger than 4GB).



**6** You'll then be asked to set your keyboard layout and create a user account. You can encrypt your personal files, but it has a performance impact.



**7** The Ubuntu files will be copied over to your hard drive, so grab a cup of tea and watch the slideshow highlighting some of the distro's features.



**8** And you're done! Once the installation is complete, your system will reboot, eject the DVD, and start Ubuntu from your hard drive. Enjoy! 🍻

# /DEV/RANDOM/

## Final thoughts, musings and reflections



**Nick Veitch** was the original editor of *Linux Format*, a role he played until he got bored and went to work at Canonical instead. Splitter!

It has been 10 years since the first version of Ubuntu launched to largely unsuspecting masses. It was brown. It was easy to install. It changed the history of Linux distros (whether or not you ever used it) and 10 years later on is more or less the most popular version of Linux in the known universe. There has been controversy. No doubt there will continue to be so. The world rarely changes without it. But even detractors would have to admit that in the primary mission of making a “Linux for human beings” it has been tremendously successful.

A lot of other things have changed in 10 years. Who would have thought you would ever see a Microsoft CEO delivering a keynote in front of a huge screen proclaiming “Microsoft ♥ Linux”? And yet, it has come to pass (<http://goo.gl/dpJloJ>). This is largely because of Azure, Microsoft’s cloud platform, which has to be Linux-friendly because various flavours of Linux (notably Ubuntu) are the most popular images used in the cloud (and about 20% of Azure).

A lot of other things haven’t changed. It is still difficult to find a choice of laptop with Linux pre-installed, and even if you decide to install it yourself, some shop staff often seem, erm, confused about what that entails (<http://goo.gl/Hnh4JO>). But wait, things have changed here too, as Currys reaffirmed that wasn’t their official policy (<http://goo.gl/U0DGCB>).

Who knows what may change in the next ten years? By the time we get around to Omnipotent Owl maybe (if *Systemd* hasn’t morphed into Skynet and destroyed humanity) people will be happy with their desktop environment, all hardware will ship with Linux support and Linux Voice will be beamed directly to your neurons. I bet Berlin still won’t have settled on a standard for office software though.



Club Mate, the essential hacker beverage.

A Yoga 13, where I boot the most recent Systemd versions on, and hence might break frequently. It’s running *Mutt* (what else!).

Connected to a fall-back desktop machine, which runs slightly less current Systemd and Fedora versions.

A Nexus 5.

Bauhaus (Mart Stam). What Bauhaus did for architecture and design might actually be a good influence when hacking ;-)

Contains memory cards, and more memory cards are lying next to the big screen, because I need to pull my pictures from my India trip off them.

## My Linux Setup **Lennart Poettering**

The creator of PulseAudio and Systemd gives us a glimpse of his coding den.

**Q** What version of Linux are you using at the moment?

**A** I have a laptop with current Fedora Rawhide, which I use for day-to-day hacking. I also have a desktop with stable Fedora 20, so that I always have a machine I can email from.

**Q** What was the first Linux setup you ever used?

**A** I got Slackware from a friend, some time in 1996, when I was 16. It was awful, I really didn’t understand a thing I was doing back then, so after two weeks I deleted Slackware again and moved back to Windows.

A year and a half later I then bought a shrink-wrapped box of Red Hat 5.0, together with a friend. I still didn’t get it really, but I found it interesting enough to stick with it, and then switched to Debian

which I stayed with for a long time.

**Q** What Free Software can’t you live without?

**A** Uh, oh. To be honest, from time to time when I travel I live without any computer or software for a month, and I really enjoy it... That said, if you press me hard I’d say *Emacs*, of course!

**Q** What do other people love but you can get on without?

**A** Hmm, good question. *LibreOffice* Impress, maybe? I do all my slides with *Latex Beamer*!

**Q** Is there a piece of proprietary software you wish were free and open source?

**A** The Coverity static analyzer is really the only thing that comes to mind here for me! ☹

# LINUXVOICE

## Systemd Cheat Sheet

| Sysvinit Command          | Systemd Command                                                             | Notes                                                                                                    |
|---------------------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| service httpd start       | systemctl start httpd.service                                               | Used to start a service (not reboot persistent).                                                         |
| service httpd stop        | systemctl stop httpd.service                                                | Used to stop a service (not reboot persistent).                                                          |
| service httpd restart     | systemctl restart httpd.service                                             | Used to stop and then start a service.                                                                   |
| service httpd reload      | systemctl reload httpd.service                                              | When supported, reloads the config file without interrupting pending operations.                         |
| service httpd condrestart | systemctl condrestart httpd.service                                         | Restarts if the service is already running.                                                              |
| service httpd status      | systemctl status httpd.service                                              | Tells whether a service is currently running.                                                            |
| service --status-all      | systemctl list-units --type service                                         | Displays the status of all services.                                                                     |
| ls /etc/rc.d/init.d/      | systemctl list-unit-files --type=service                                    | Used to list the services that can be started or stopped. Used to list all the services and other units. |
| chkconfig httpd on        | systemctl enable httpd.service                                              | Turn the service on, for start at next boot, or other trigger.                                           |
| chkconfig httpd off       | systemctl disable httpd.service                                             | Turn the service off for the next reboot, or any other trigger.                                          |
| chkconfig httpd           | systemctl is-enabled httpd.service                                          | Used to check whether a service is configured to start or not in the current environment.                |
| chkconfig --list          | systemctl list-unit-files --type=service<br>ls /etc/systemd/system/*.wants/ | Print a table of services that lists which runlevels each is configured on or off.                       |
| chkconfig httpd --list    | ls /etc/systemd/system/*.wants/httpd.service                                | Used to list what levels this service is configured on or off.                                           |
| chkconfig httpd --add     | systemctl daemon-reload                                                     | Used when you create a new service file or modify any configuration.                                     |

| Sysvinit Runlevel | Systemd Target                                        | Notes                                                                                        |
|-------------------|-------------------------------------------------------|----------------------------------------------------------------------------------------------|
| 0                 | runlevel0.target, poweroff.target                     | Halt the system.                                                                             |
| 1, s, single      | runlevel1.target, rescue.target                       | Single user mode.                                                                            |
| 2, 4              | runlevel2.target, runlevel4.target, multi-user.target | User-defined/Site-specific runlevels - identical to 3.                                       |
| 3                 | runlevel3.target, multi-user.target                   | Multi-user, non-graphical. Users can usually login via multiple consoles or via the network. |
| 5                 | runlevel5.target, graphical.target                    | Multi-user, graphical. Usually has all the services of runlevel 3 plus a graphical login.    |
| 6                 | runlevel6.target, reboot.target                       | Reboot                                                                                       |
| emergency         | emergency.target                                      | Emergency shell                                                                              |

| Command                                             | Notes                                                                            |
|-----------------------------------------------------|----------------------------------------------------------------------------------|
| systemctl get-default                               | Determine which target unit is used by default.                                  |
| systemctl set-default multi-user.target             | Change default boot target to multi-user.target.                                 |
| journalctl -b                                       | Show all messages from this boot.                                                |
| journalctl -b -p err                                | Show all messages of priority levels ERROR (4) and worse, from the current boot. |
| journalctl -p warning --since="2014-06-14 23:59:59" | View the warning or higher priority messages from certain point in time.         |
| journalctl -f                                       | Follow new messages.                                                             |
| journalctl /usr/sbin/httpd                          | Show all messages by a specific executable.                                      |
| journalctl --full                                   | Display full (= not truncated) messages.                                         |
| systemctl --state=failed                            | Lets find the systemd services which fail to start.                              |
| systemctl list-units --type=target                  | Show current runlevel.                                                           |
| systemctl isolate graphical.target                  | Changes the current target (runlevel).                                           |
| systemctl rescue/emergency                          | Changing to Rescue(single user mode)/Emergency Mode.                             |
| systemd-cgls                                        | cgroup tree                                                                      |
| systemctl show -p "Wants" multi-user.target         | What other units does a unit depend on?                                          |
| systemctl list-jobs                                 | Check for possibly stuck jobs use.                                               |

| Old Command       | Systemd Command        | Description                         |
|-------------------|------------------------|-------------------------------------|
| halt              | systemctl halt         | Halts the system.                   |
| poweroff          | systemctl poweroff     | Powers off the system.              |
| reboot            | systemctl reboot       | Restarts the system.                |
| pm-suspend        | systemctl suspend      | Suspends the system.                |
| pm-hibernate      | systemctl hibernate    | Hibernates the system.              |
| pm-suspend-hybrid | systemctl hybrid-sleep | Hibernates and suspends the system. |

# Matrix

## Arm Mini PC



- TBS TUNER SUPPORT
- FREESCALE QUAD CORE
- 100% OPEN SOURCE
- XBMC, VDR, TVHEADEND



NOW WITH  
**openelec**  
EMBEDDED LINUX ENTERTAINMENT CENTER

NOW ONLY  
**£119**

FREE UK MAINLAND DELIVERY

|                         |                                                     |
|-------------------------|-----------------------------------------------------|
| System-on-a-chip (SoC)  | Freescal MCIMX6Q5EYM10AC                            |
| CPU                     | Quad ARM Cortex-A9 at 1.0GHz                        |
| GPU                     | Vivante GC2000, Quad core GPU, Quad IPU             |
| RAM                     | 2GB DDR                                             |
| USB 2.0 ports           | 3x USB 2.0                                          |
| Audio & Video interface | HDMI port<br>3.5 mm jack                            |
| Storage                 | eMMC 16GB<br>1x SD card slot<br>1x TF card slot     |
| Network                 | 10/100/1000 wired Ethernet<br>WIFI IEEE 802.11n/b/g |
| Power input             | 5V, 3A                                              |



BY YOUR VERY OWN: BEN EVERARD

SOUNDS TOO GOOD TO BE TRUE? IT'S NOT!  
UNLIKE OTHER XBMC SOLUTIONS, OPENELEC IS NOT BASED ON UBUNTU.  
IN FACT, IT'S NOT BASED ON ANY LINUX DISTRIBUTION;  
OPENELEC HAS BEEN BUILT FROM SCRATCH SPECIFICALLY TO ACT AS A MEDIA CENTER.

"XBMC performance is fantastic and could make the Matrix one of the best frontends you could buy"

"The hardware looks good, feels solidly made and works well"



**xbmc**



**WWW.TBSCARDS.CO.UK**