

LINUX VOICE

The magazine that gives back to the Free Software community

July 2014

RASPBERRY PI WILDLIFE

Spy on badgers with the Pi's camera module

SERVER HARDENING SECURITY

Keep one step ahead of the script kiddies

CUSTOM HARDWARE FPGAs

Program a sound chip for crunchy 8-bit bleeps

**115 PAGES
OF NEURAL
ENHANCEMENT!**

UBUNTU 14.04 RESPINS

The world's biggest distro has spawned a new breed – find your favourite!



ACER C720 Chromebooks: now officially a great laptop
CODE CLUB Teach the world to code, one child at a time
JOHN VON NEUMANN The man who invented your processor

RASPBERRY PI

ROBOTICS

Make a start on your cyborg army



GAMES

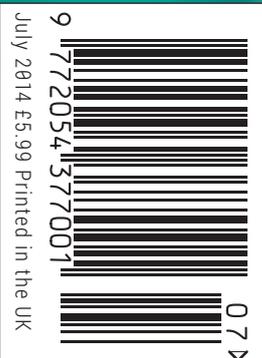
STEAM OS

Install it now and level up your Linux gaming



34+ PAGES OF TUTORIALS

FREE SOFTWARE | FREE SPEECH



July 2014 £5.99 Printed in the UK

ISSN 2054-3778

07D

As the stewards of the Open Source Definition (OSD) and the community-recognized body for reviewing and approving licenses as OSD-conformant, the OSI facilitates Open Source community-building, education, and public advocacy to promote awareness, adoption and the importance of non-proprietary software.



The OSI, with global reach, champions Open Source software and projects, meeting with developers, users and communities as well as with executives from the public and private sectors to explain how Open Source technologies, licensing & models can provide economic, strategic and societal advantages.

Open Source Initiative welcomes Linux Voice to the global Open Source community.

Join the Open Source Initiative now
and be a part of the future of Open Source.

[<opensource.org/members>](http://opensource.org/members)



ELECTRONIC FRONTIER FOUNDATION

Help EFF Defend Your Rights in the Digital World eff.org/join

Linux for human beings

The **July** 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 (you again).

THE LINUX VOICE TEAM

Editor Graham Morrison
graham@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Technical editor Ben Everard
ben@linuxvoice.com

Editor at large Mike Saunders
mike@linuxvoice.com

Games editor Liam Dawe
liam@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Mark Crutch, Josette Garcia, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Jonathan Roberts, Mayank Sharma, Valentine Sinitsyn



GRAHAM MORRISON

A free software advocate and writer since the late 1990s, Graham is a lapsed KDE contributor and author of the Meeq MIDI step sequencer.

Where would we be without Ubuntu? Even if you don't agree with the direction it's taken in recent years, there's very little doubt in my mind that Linux would be much, much worse off had Mark Shuttleworth not decided to spend a proportion of his Thawte windfall going to space creating the Ubuntu Foundation. It transformed public expectation of what Linux was and could be, making it easy to install and making Ubuntu as close to a standard for commercial and hardware vendors as possible. This has had a positive effect on all Linux distributions.

Anyone can build on the good work of both the Debian and Ubuntu teams to create their own distributions, and it's this initiative and diversity that we're celebrating in issue 4. Each Ubuntu derivative has a unique reason for being different, and the success or failure of these derivatives depends entirely on how effective those distributions are at communicating and implementing their ideas. I can't think of a more democratic system for success, and it only happens with Free Software.

Graham Morrison
Editor, Linux Voice

**SUBSCRIBE
ON PAGE 60**

What's hot in LV#004



ANDREW GREGORY

Code Club is having a very real and positive effect on how IT is taught to children in the UK. This makes me happy. **p30**



BEN EVERARD

It's wonderful to be able to welcome Chris Brown to Linux Voice, and he kicks off with an ace guide to processes. **p64**



MIKE SAUNDERS

Ben is very much keeping the spirit of Colin Pillinger alive with his own Mars Rover project made from an ice cream carton! **p78**



CONTENTS

July LV004

Beauty is truth, truth beauty. That is all ye need to know.

18

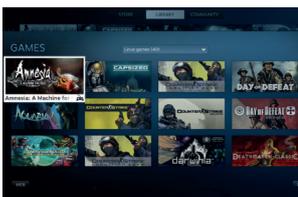
UBUNTU

14.04 RESPINS

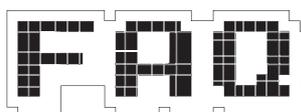
30

CODE CLUB: THE NEXT GENERATION

Inspire the kids of today to achieve great things. You can be their Yoda!



34 STEAMOS
How Valve is shaking up the world of games.



38 FAQ LLVM/Clang: the future of compiling code on Linux.



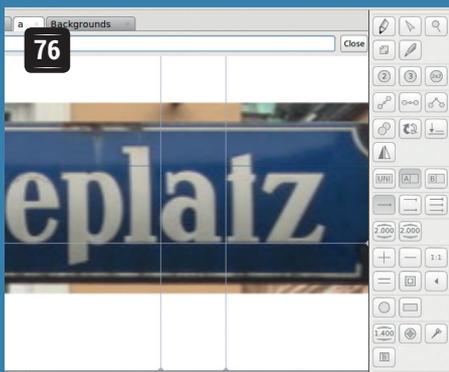
26 KANO
Make computers as easy as Lego.



REGULARS

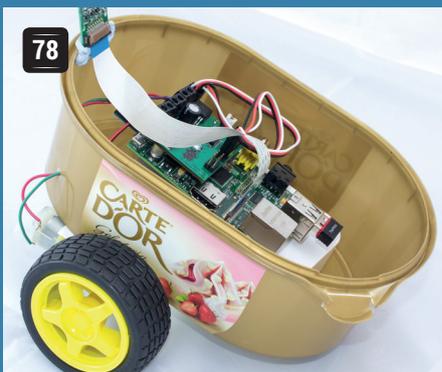
- 06 News**
In the wake of Heartbleed, OpenSSL forks. DON'T PANIC!
- 08 Distrohopper**
What's going to be on your Linux box in 2015.
- 10 Gaming**
The latest AAA-grade titles are coming your way!
- 12 Speak your brains**
Let us know what's going on in your precious grey matter.
- 16 LV on tour**
Epistles from Lisbon, Dubrovnik and Lincolnshire.
- 40 Interview**
We visit England's Maker Belt to speak to Pimoroni.
- 54 Group test**
The best terminal emulators to help you Get Stuff Done.
- 62 Sysadmin**
Turn your ever-waking eye on the logs. Intruders, fear me!
- 64 Core technologies**
Linux Voice welcomes a new writer: Dr Chris Brown.
- 68 FOSSpicks**
Only the hottest free software makes it into these pages.
- 102 Code ninja**
Error correction: because only Ben Everard is perfect.
- 110 Masterclass**
Media players of the graphical and command line flavour.
- 114 My Linux desktop**
We peek into the control room of gamingonlinux.com.

TUTORIALS



Make your own typeface with BirdFont

Design software has never been more accessible. Now go and brand something!



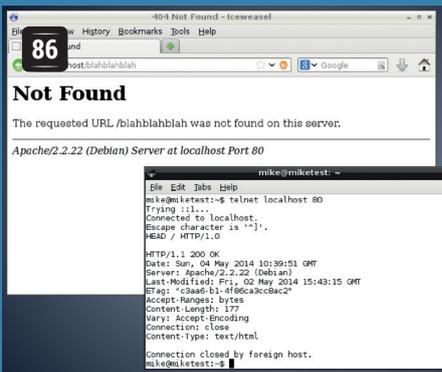
Raspberry Pi: Build a Mars Rover

Create robots programmable in Python quicker than you can say "Sarah Connor".



Raspberry Pi: Monitor woodland creatures

Keep tabs on your local badger population with a remote, home-brewed wildlife camera.



SSH, Apache & Tiger: Secure your servers

Stay one step ahead of the script kiddies who want to vandalise your web servers.



VirtualBox: Keep Windows XP after migration...

... should you wish to, that is.



John von Neumann: EDVAC & IAS

The Manhattan project bore strange fruit...

98 FPGAs: Build your own custom sound chip

Dip your toe into chip design

104 Android: Develop your first application

Fun and frolics on a free phone OS.

REVIEWS



46 Acer C720 Chromebook

With one hand Google gives excellent hardware. With the other it takes our privacy...



48 Ubuntu 14.04

The latest Long Term Support version of the world's biggest distro has quietly enthused us. Shhhh!

49 TBS Matrix

An ARM box that doubles as a smart TV device, running XBMC and Tvheadend. Sounds good to us!



50 BitScope BS10

Electronics tinkerers, rejoice: you have nothing to lose but your ignorance of current through a given component.

52 Books

Fresh reading matter to further enhance your neurons.



NEWS ANALYSIS

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

Opinion

Lessons for Open Source from Heartbleed

The armageddon has been and gone – let's not waste a good crisis



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

We've had a while for the shock of the Heartbleed announcement to sink in and there's a lot to consider. While the first impressions might be about the serious, exploitable bug and the repercussions of its abuse, the incident casts light on both the value and risks of open source. All programmers make mistakes, so there's no huge surprise that it happened in OpenSSL. But why did it go undetected for so long? There are several contributing factors.

It's hard to spot error in complex code:

First, the OpenSSL code is huge, years old and implements a set of algorithms that need specialist cryptographic knowledge to understand them. Reading someone else's code in this context is difficult, dull and time consuming. That's not a recipe for scrutiny even with a large paid team.

Community too small: There is no large, paid team. The whole community developing and maintaining OpenSSL was no larger than 11 people before Heartbleed, with only one of them working full-time on the code. Despite being widely deployed, the community did not receive regular participation from developers using

OpenSSL in other projects. There's been speculation why. An obvious explanation is that the cryptographic complexity of the code meant that non-specialists were not effective as community participants, but US government open source specialist David A Wheeler has speculated that the unusual licensing arrangement for OpenSSL – a custom open source licence with non-GPL-compatible copyleft effects and potentially challenging advertising clauses – also discouraged community involvement.

Exploit detection turned off: What can a small community do to spot uninitialised memory and buffer over-run errors? String handling libraries today often include detection at compile-time or even run-time for such errors, and the OpenBSD code used by OpenSSL for this purpose is no exception. But according to Theo de Raadt, leader of the OpenSSH project, these detection capabilities had been turned off at compile time in the OpenSSL build for performance reasons long ago and had never been turned on again, despite the performance issues being addressed in the code.

Would proprietary be better?

Doing this with proprietary code would be unlikely to make things better. Hiding a development team behind NDAs and corporate secrecy, having their priorities driven by unseen managers and keeping code invisible to potential users all constitute an anti-pattern for security software. In addition, the ability of open source to bring all the best hands to the problem once it's identified would simply not

exist with a proprietary solution. Engaging would need permission and bureaucracy, and many contributors would just say no.

Should I make donations?

So what's the right way to react? Fork the code? No, that's been done, and probably just increases the problem by removing potential expert participants from the OpenSSL pool. Re-implement it? That's been tried as well. Even with various forks and re-implementations, OpenSSL remains widely used, because the problem is complex and the experts are few. Those new projects are unlikely to surpass in a few months what OpenSSL has largely succeeded at doing over a decade. Donate money? While cash donations to a project can be a short-term fix, in the long term it is unlikely to help unless it leads to more developers both writing and testing the code. The best fix would be for the companies most dependent on OpenSSL to hire experts and pay them to join the community and work on the code.

This, after all, is the key to open source. It's not about free stuff; rather, open source delivers the liberties that enable developers with differing motivations and origins to collaborate on a codebase without needing to ask permission from anyone other than each other, and even then only out of social effectiveness. Throwing money at an open source project doesn't automatically make anything better; that takes people with actual skills.

Heartbleed has shown us that open source is no guarantee of invulnerability. Fortunately, the crisis has highlighted a set of needs that are being met in a way no other approach would have allowed. Looking past the crisis, it's possible Heartbleed is actually making things become better, faster.

“Throwing money at an open source project doesn't automatically make anything better.”

CATCHUP

Summarised: the eight biggest news stories from the last month

1 OpenSSL has been forked: say hello to LibreSSL

After all the fun and games of the OpenSSL Heartbleed vulnerability, which made vast swathes of the internet open to cracking, the OpenBSD team (see news item 8) has had enough. A bunch of developers has forked OpenSSL into LibreSSL, and they're going through the code discovering some truly horrendous holes and coding errors. Right now the website is in Comic Sans with blink tags, but donations should fix that: www.libressl.org

2 Linus Torvalds wins yet another award

Not content with having won the Technology Academy of Finland's Millennium Technology Prize, NEC's C&C Prize, the Takeda Award for Social/Economic Well-Being, the British Computer Society's Lovelace Medal and the Electronic Frontier Foundation Pioneer Award, everyone's favourite narky kernel hacker has another to add to the list: the 2014 IEEE Computer Society Computer Pioneer Award. For his contributions to computing, Linus can expect a bronze medal in the post.

3 Firefox 29 sports a redesigned interface

This has been massively controversial, but then any major UI update for a well-used app always is. Firefox 29 features a trimmed-down interface that "makes it easy to focus on web content". In addition the redesigned tab bar is "sleek and smooth to help you navigate the web faster". Sounds a bit buzzwordy to us... www.mozilla.org



4 Canonical puts Ubuntu for Android project in limbo

Ubuntu maker Canonical is ambitious with its plans to have the Unity interface everywhere: on phones, tablets, TVs and desktops. One part of this convergence strategy has been kicked into the long grass though: Ubuntu on Android. This was supposed to provide a regular Android experience on your mobile devices, but when you plugged them into a TV or monitor, you'd get the full Unity experience. Canonical has said it isn't dead, but they'll focus on Ubuntu phones for now.

5 Band releases album as a Linux kernel module

This is charmingly silly. Netcat, a Seattle-based band that "explores the intersection between technology, complexity and free improvisation" has just kicked out a new album. But listening to it is a challenge: you have to build a custom kernel module that creates a `/dev/netcat` device node that you can then pipe into an audio player. But be warned: if you decide to go this far, you'll need "several gigabytes of memory" to build it. See: <http://tinyurl.com/netcatmod>

6 It's official: Edward Snowden is a Linux user

Many people suspected this all along, but now it has been confirmed that leaker extraordinaire Edward Snowden used Linux in his data gathering antics. Specifically, he used the Tails distribution, a highly paranoid flavour of Linux that routes all of its internet traffic through Tor and stores everything in RAM (thereby leaving no trace of activity on a computer's hard drive). We still don't know if he prefers Vim over Emacs though. Tails' site: <https://tails.boum.org>

7 New consortium that includes Microsoft to fund core FOSS projects

More fall-out after the Heartbleed fiasco: a bunch of big-name companies including Amazon, Dell, Facebook, Google, IBM, Intel and Microsoft have teamed up with the Linux Foundation to offer financial support for "critical open source projects". And guess which project will receive funds first? Yes, OpenSSL. Finally, the big companies that use FOSS realise it doesn't grow on trees. <http://tinyurl.com/lrlmzh4>



8 Ultra-secure Unix flavour OpenBSD 5.5 released

Hang on – this isn't Linux! True, but we like to support our other FOSS brethren in the operating system world too. OpenBSD is a lot like Linux, offering oodles of advanced security features but not so much desktop hardware compatibility. It's superb for small servers and routers though. The 5.5 release brings about a 64-bit `time_t`, which means that the operating system's clock won't overflow until Sunday, 4 December in the year 292,277,026,596. www.openbsd.org

DISTROHOPPER

We've tapped GCHQ's communications to find out what's going on in distro land.

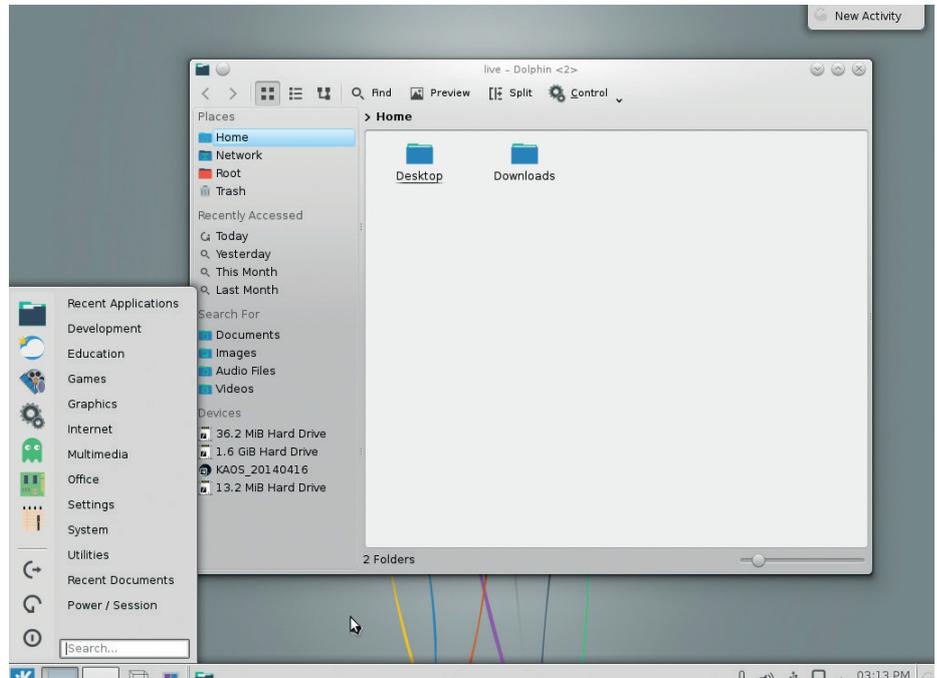
KaOS

A simple and elegant KDE.

KaOS is about providing a simple, clean KDE experience. In order to achieve this, the developers have slimmed the repositories to just over 2,000 packages. This is a minuscule number by most distros' standards, but it provides everything most people will need.

By reducing the amount of software available, KaOS creates a consistent environment containing only what the distro's creators think are the best applications for each task. In some cases it enables you to deviate your environment from pure Qt, so Firefox and LibreOffice are available (though not installed by default).

The KDE interface is quite different from the standard, and the focus seems to be on keeping things simple. The developers have managed to build a unique and slick desktop without getting sucked into the pitfall of adding too many graphical effects – a trap that's all too common on KDE distros. Our only criticism of the user interface is the lack of tooltips on the menu. This may make it a



We can't think of a less appropriate name than KaOS for such a refined KDE experience.

little difficult for new users to get to know the system. It is, however, a minor gripe on an otherwise excellent interface.

Ultimately, we enjoyed experimenting with KaOS, but we would struggle to use it

full-time, as there just isn't enough software for us. It's not aimed at us though – it's intended to be a lean distribution, and that's what it is, as well as being probably the best-looking KDE distro we've used.

Deepin

Deepin is more than just a great software centre.

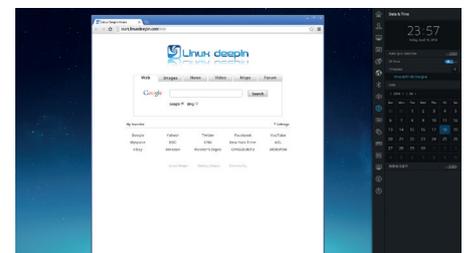
The English-speaking Linux world has been a little slow to cotton on to the fact that Deepin has been a fantastic distro for a few years now (it's Chinese). It was among the first to feature a software centre for finding, reviewing, and installing software. The Deepin Software Centre has always been one of our favourite apps in this area, and the newest version is no exception.

Deepin has its own desktop environment built on web technologies, and it looks great. The best feature for us was the settings menu. This slides out of the left-hand edge

of the screen, a little like the way that Android settings slide down from the top.

The biggest let-down for this reviewer was the Launcher (similar to Gnome's Dash). It spread the application icons out too thinly and made it take too long to find what we were looking for. Additionally, we wouldn't recommend Deepin for lower-powered machines, as the performance is just a little on the slow side.

We really liked the attention to detail in some areas. For example, the home screen of the web browser (Chrome) has a nice



The slide-out settings and browser home page are both great, innovative features.

setup for easily selecting to search in Google or Bing for websites, videos, places, etc.

Deepin deserves more praise than it gets for producing some cracking pieces of software. If it could just sort out that pesky launcher, it would be a top distro.

Makulu

A quirky distro for quirky people.

Makulu comes with a number of different desktops, including the most recent release of Mate. All of them are heavily themed with cartoon-y icons and earthy colours, which give the distro a pleasant, informal feel. Underneath all this, Makulu is built on Debian, so there is of course all the usual underlying Debian software (plus a host of other repositories including the ones from Mint, SolydXK, Opera, Skype, various Google projects, Mate and its own ones). In other words, there's loads of the latest software available.

On launching Makulu, the first thing we noticed was the office suite: Kingsoft. You may not have heard of this, but it's been around for a while (though the Linux version is new. In fact, it's still considered in Alpha). It's not open source, but it is free (no cost) for the basic version. The most noticeable

“The Makulu developers have made some bold choices, and that's refreshing to see.”



Makulu means 'chief' in Zulu – a fact that helps explain the distro's styling.

thing about Kingsoft is that it has an interface similar to MS Office's ribbons. This is a little controversial, but it could help newer Windows refugees feel at home. For a Linux user, a far more serious criticism of this office suite (other than the fact that it's closed source) is that it doesn't support ODF files.

The web browser is Chrome, but with a number of extensions to make it a little more friendly. Adblock Plus is probably the most notable of these, but there

are also notifications for most popular social networks.

Makulu isn't as polished as some distros, and its strong personality will put some people off, as will the inclusion of non-free (as in speech) and social media software. However, the developers have made some bold choices, and that's refreshing to see. If you're bored with mainstream distros and looking for something a bit new and fresh, Makulu is worth a look, if only to see what can be done when a development team tries something a little out of the ordinary.

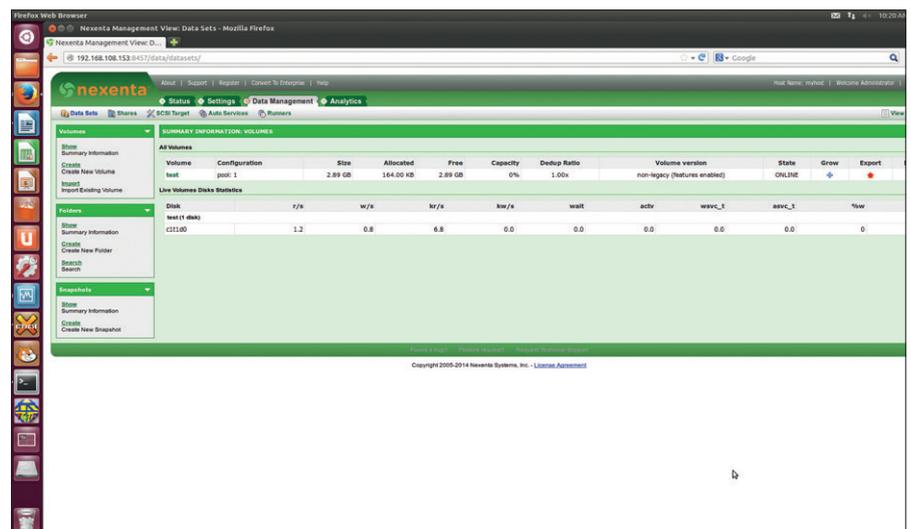
NexentaStor 4.0 Where ZFS feels at home

NexentaStor isn't Linux. It's not even a BSD. It's based on the Illumos kernel, which is derived from Solaris (via OpenSolaris), even further back in the Unix family tree. This is the branch of Unix where ZFS started life. This filesystem is the distro's reason for living. In fact, the heart of NexentaStor is a web interface for managing ZFS data stores.

For this purpose, it's fast and hugely powerful. It's perhaps better suited to managing enterprise storage solutions than setting up a home NAS system, but that doesn't mean that you can't use it for either task.

After freely downloading distros of Linux and BSD, it felt a little weird to have to register for a licence to use NexentaStor. It is free (as in beer) for the community version, but it doesn't feel very FLOSS-y to have to type in a licence code in order to get it to work.

Once you get past this, there's a powerful HTML interface for managing and configuring a mind-boggling array of options that remind you that it's not a plug-and-play system for quickly adding a backup server to your home network, but an enterprise-ready storage system for handling networked computers storing terabytes of information. To this end, it's



Network storage isn't everyone's idea of a good time, but it can be quite an eye opener to see all the options laid out in NexentaStor's absurdly powerful HTML interface.

designed to integrate well with OpenStack, the enterprise cloud platform that all big businesses love these days. If you're interested in learning

about the ZFS filesystem, then using it on its native kernel is a good place to start. Likewise, if you're managing a data centre, it's a useful option.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons



GNAEUS JULIUS AGRICOLA



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

Wine enables us Linux users to run Windows applications under Linux without the need for a Windows licence. It sounds great, but it does come with its own set of drawbacks, such as poor performance in certain games; and some games may refuse to work without a lot of tinkering.

A problem has arisen now that Linux is gaining a foot-hold in the gaming market: "Wine Ports", developers who bundle their game with a copy of Wine and call it a day. I know what you're thinking: "That's not a Linux port!", and you are right. It seems some developers are trying to short-cut their way onto Linux like they do on Mac OS X.

Sometimes bottling Wine along with a Windows game can be acceptable, if, for example the source code to an older title has been lost; we can live with that. This situation is good because of two things: it gives us a game we wouldn't otherwise get, and it counts as a Linux sale to the developers to show them interest for futures titles to be ported.

The problem is when developers look to cut corners and shoehorn a Linux version out the door without looking into porting it natively. This can be a problem, as these developers could see poor sales due to using Wine and Linux users being wise to that fact, the developers can then easily claim Linux doesn't sell.

So I ask you dear readers, how do you feel about Wine Ports? Let us know on the Linux Voice forum. <http://forums.linuxvoice.com>

Star Conflict

Like being a real space pilot, without the risk of death.

Big news space fans! Star Conflict, a free-to-play spaceship combat MMO has officially launched its Linux version as promised. This is one of our first major free-to-play titles that's been released from outside Valve.

The graphics are simply fantastic, which is surprising for a free game. It also doesn't shove anything into your face about paying for in-app purchases – it's all perfectly optional, which is again very surprising. You can buy extra credits for ships and paint jobs or buy downloadable content (DLC) packs from Steam.



The game starts you off slowly to not overwhelm you with a few tutorials showing you basic flight and combat, which it makes really painless for you.

This is the first Linux game from Star Gem Inc, and we hope it will only be the start for them. See you in space! <http://store.steampowered.com/app/212070/>

Counter Strike: Global Offensive

The next-generation of what was originally just a mod.

Do you see yourself as a counter-terrorism hero? Or do you see yourself as an evil terrorist? Well now you'll get the opportunity, as Counter Strike Global Offensive is coming to Linux. The news comes as a result of the Reddit "Ask Me Anything" that Gabe Newell (the owner of Valve and Steam) did recently. There is no ETA, but Gabe assured us that it is coming. With Valve's notorious "Valve Time" it could still be a while before it comes to our beloved platform, so we just have to be patient for a little while longer.

CS:GO is a next-generation police vs terrorists first-person



shooter boasting improved graphics, new game modes, support for weapon customisation and a lot more.

One of the best parts of CS:GO is the frantic and fun new mode called "Gun

Game" where you swap guns on each kill. The last weapon is a knife, which doesn't make it easy, but it does make it interesting... <http://store.steampowered.com/app/730/>

Luftrausers

Take on the Nazis in a crazy-coloured world.

Luftrausers is an arcade-style aerial combat game with some over-the-top action from the well-respected developer Vlambeer.

The game features over 125 combinations of weapons, bodies, and propulsion systems offering you a lot of customisation to keep you coming back for more. It also opens up different ways to play each level as you try different combinations to beat it, you can unlock different colour-sets to play the game in to keep you interested.

The game won't bring your PC to a crawl either, so it should work on some older computers. As it uses so little in the way of system resources, you only need a graphics card that has OpenGL 2.1, which is quite old now.

Luftrausers is a great timewaster for the armchair general to sit back comfortably with a game-pad in hand and annihilate some enemy planes, submarines, ships



Luftrausers' simplistic graphics and frantic gameplay remind us of 80s arcade machines.

and more. It lacks the glitz of a modern FPS, but it's full of charm.

<http://store.steampowered.com/app/233150/>

Unvanquished

You might want to keep one eye to the ceiling in this one.



Unvanquished is a hybrid of first-person shooter and real-time strategy wrapped in a good-looking package for open source fans. The game is now in its early alpha stages, but it is already playable and looking really slick. It pits human marines versus vicious aliens in all-out war. The game improves dramatically with each new release, and the team are always looking for help with programming and translations for anyone to get involved.

The game is a fork of another open source game, named 'Tremulous', which sadly seems to have died out, but the survival of Unvanquished is what makes open source great.

www.unvanquished.net

Xonotic

The direct successor of the Nexuz first-person shooter project.

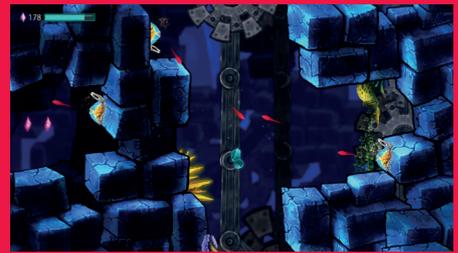


Xonotic is another free and open source shooter; this time, rather than pitting humans versus aliens, it's war between humans in fast-paced game modes. Xonotic is similar to the famed Unreal Tournament series of games only with no price tag attached to it. It features many weapons, each with their own secondary fire mode, and there are plenty of game modes to keep you going. If lots of weapons aren't enough for you it also has 18 different maps, all wrapped up in a neat free package.

The game has its own statistics system integrated with it, so you can even keep track of how terrible you are!

www.xonotic.org

ALSO RELEASED...



Beatbuddy: Tale of the Guardians

Your ears are in for a real treat with Beatbuddy: Tale of the Guardians. The game is an exploration/puzzle game that involves music in everything you do, with fantastic results. The levels are all hand-painted giving it that extra loving touch, and it really is a beautiful-looking game. Stop the evil Prince Maestro from stealing all the music!

<http://store.steampowered.com/app/231040>



Unreal Engine 4

Epic Games has unleashed hell on Linux as it has announced that Unreal Engine 4.1+ will support Linux officially. Unreal Engine is an extremely popular engine with big-name developers, so this is some of the best news we have had since Steam on Linux. Epic is offering the engine at a low price with source code access too.

www.unrealengine.com/blog/41-update-preview



CryEngine

Crytek (the developer behind such games as Crysis) has ported its CryEngine games engine to Linux. This means in future we will be able to see games as graphically amazing as Crysis on Linux, and who doesn't want that to happen? The free SDK is looking like it will add Linux support too, so onwards and upwards from here for all of us.

www.crytek.com

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

LOOK! IT'S A BABY READING A MAGAZINE!

Hello guys. I hope you're settling into the new digs. OK, great start on the new magazine. I'm writing to suggest a feature article, after a recent addition to my family got me thinking about parental controls. It's quite a broad topic with lots to write about yet there seems to be a lack of people out there writing good how-to articles on it. So my question to you is: Can you consider writing a beginner friendly article on how to protect children on their computers, breaking the jargon down into easy to understand segments? The main thing I would like to know about is, how to filter out mature content for

children whilst allowing the grown ups unrestricted use.
Mark Skinner
 PS. I've attached a little snapshot of the Linux trainee, Jacob, reading his favourite magazine. You might think I'm getting ahead of myself, but there's no such thing as being too prepared, right?
Mike says: There's a lot of parental controls built-in to Linux through the permissions system. In a large organisation a sysadmin can control what the users do on a machine, and likewise a parent can use the same system to limit (for example) access to the internet. We'll look into a tutorial and present our findings.



Parental advisory – keep children away from the beer brewing tutorial with Raspberry Pi on page 76. Alcohol is bad for you, kids!

KERNEL HACKING

I picked up my first issue of Linux Voice [LV002] and was happily reading through it when I came upon Dr Sinitsyn's article on kernel modules. What a wonderful piece; clear, concise but still comprehensive.

Please, please give us more articles like this – it was a wonderful gateway into the most fascinating part of the Linux experience: the ability to tinker with the kernel.

Pedro Pinto

Andrew says: You'll be pleased to learn that we have at least two more tutorials in the pipeline from Dr Sinitsyn, on programming USB devices and using PyParted. There's not much documentation for PyParted out there at the moment, so we're going to release that one as CC-BY-SA as soon as we've published it, to enable more people to bask in his awesomeness. And to everyone who missed the kernel tutorial, we've just put it up on www.LinuxVoice.com for your perusal.

LV CODING KERNEL

LINUX VOICE
TUTORIAL

VALENTINE
SINITSYN

WHY DO THIS?

- Understand the common features shared by all kernel modules.
- Earn brownie points in job interviews.
- Tinker with the foundations of your Linux machine.

LINUX KERNEL: YOUR FIRST MODULE

Ever wanted to start hacking the kernel? Don't have a clue how to begin? Let us show you how it's done...

Kernel programming is often seen as a black magic. In Arthur C Clarke's sense, it probably is. The Linux kernel is quite different from its user space: many abstractions are waived, and you have to take extra care as a bug in your code affects the whole system. There is no easy way to do floating-point maths, the stack is fixed and small, and the code you write is always asynchronous so you need to think about the concurrency. Despite all of this though, the Linux kernel is just a very large and complex C program that is open for everyone to read, learn and improve, and you too can be a part of it.

Probably the easiest way to start kernel programming is to write a module – a piece of code that can be dynamically loaded into the kernel and removed from it. There are limits to what modules can do – for example, they can't add or remove fields to common data structures like process descriptors. But

LXR (Linux Cross Reference) is a web-based system for navigating Linux sources (and other large codebases).

Our introduction to kernel hacking took a scary subject and made it accessible, as you can see over at www.linuxformat.com.

BAD DANGER

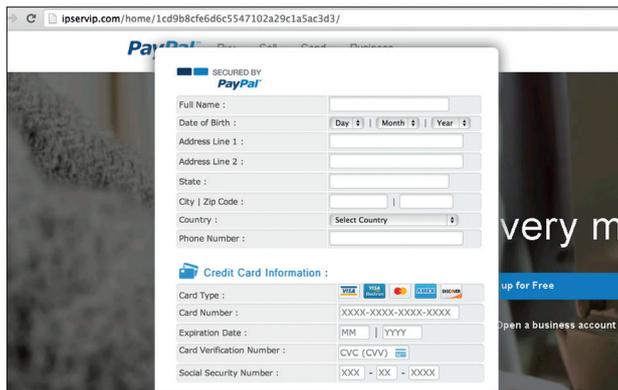
My credit card was blocked today due to suspicious charges on my bill and the agent at my credit card company suggested that I upgrade my anti-virus and firewall. He thought that my card might have been compromised as I made some online donations three days before the odd charges. I made the donations using Iceweasel 24.4.0, and I can't say that I checked to make sure there was an SSL/TLS lock on the websites when I made the donations.

Is there anything I could have done to provide greater security for myself during the donations, barring double-checking the security lock? I'm using Debian Testing and I'm not running an anti-virus or firewall.

Roy Birk, Maryland, USA

Ben says: Ultimately, there are three parts to the security of an online transaction: that part on your computer, the part where the data is sent and the part where the data is on the recipient's computer. Each of them is secured in different ways.

Most Linux distros are pretty secure by default, and there are no real antivirus software, and don't know any Linux users that do. Although you say you're not running a firewall, you almost certainly are (unless you've deliberately disabled it), as



there's one built into the Linux Kernel. Simply keeping up with the latest software in your distro's repositories should be sufficient in most cases, although you'll have to take a few more precautions if you run any public servers on the computer. You probably shouldn't do internet banking on a computer that's running a public server anyway.

The second part is when the data is in transit. Here, it really is essential to check that the site is being served over SSL (this will give the lock icon in the browser's URL bar). If there are any browser warnings at all, you should be very suspicious of the site.

Finally, there's the part where it's on the other company or organisation's servers. Once you send your credit card details, they will then store and process them. If their servers have been compromised, then your details could be lost. Unfortunately, there's not a lot you can do about this other than only sending your details to organisations you trust.

Not every dodgy site is as blatant as this fake PayPal verification screen.

CODE WANTED

I'm a long time Linux user and FOSS advocate, and I'm about to start a small business that will need a custom application written. Unfortunately, I can't code. What's the best way to hire someone that's active in the FOSS community to write and maintain an application, including managing things like (potential) code contributions?

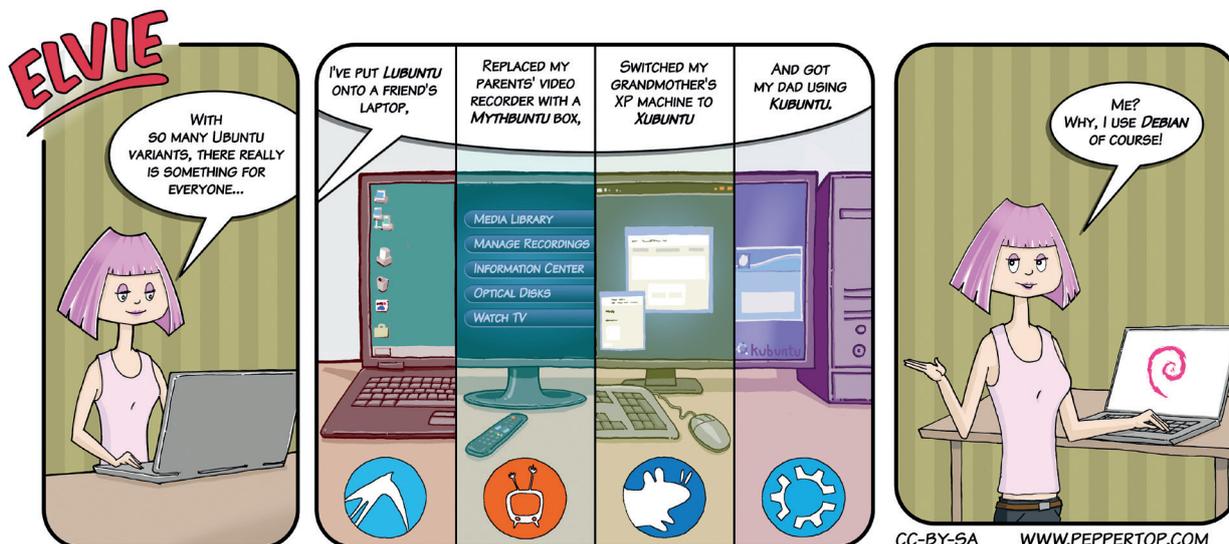
Noah

Andrew says: First of all, you're asking in the wrong place: forums such as forums.linuxvoice.com are much better for getting a question in front of many eyeballs and many brains quickly, and getting a prompt answer.

Second, you may want to look in the direction of www.linuxrecruit.co.uk. There's a vacancy advertised on there at the time of writing for a System Administrator at £400 a day, with essential skills listed as Amazon Web Services, Puppet, Red Hat/centOS, Jenkins and designing scalable infrastructure – all of which we've covered in Linux Voice.



Enhance your neurons as well as your job prospects by learning Linux.



CC-BY-SA WWW.PEPPERTOP.COM

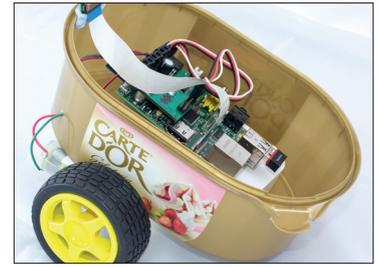
GIVE YOUR DATABASE AN INFORMATION IMPORT.

We'll share our knowledge to improve yours.

100% of our clients rate our PostgreSQL training courses as excellent. Book your place and gain access to unrivalled knowledge of the core code.

FEATURE REQUESTS

Greetings from Down Under! Well don on avoiding the patronising "Point here; click here; to save the file, go to File, then click on Save" mentality that you see elsewhere. I certainly don't want to sit down to read a magazine only to be told things that I could have worked out for myself, as I consider myself to have at least half a brain. I managed to install Linux didn't I, and that was before Mandrake came along and made it easy, and before Ubuntu came along and made everything work. The comet hunting tutorial in LV003 was



An army of killer robots await, powered by Linux and Raspberry Pi.

brilliant, but I want more... give us robots, give us supercomputers – just keep it coming!

Andrew says: Thanks Sarah – just for you there's a Mars rover-style botanics tutorial on page 78.

SPENDING MONEY FOR FUN

I'm completely baffled by the UK government's decision to pay Microsoft £5.5m to prolong support for Windows XP. It's puzzling on so many levels...
1/ Why are so many government machines still running XP in the first place? We knew when support was going to end. If it were my money I'd make damn sure that I had a solution in place well before time. Oh wait, it is my money...
2/ What are these XP machines being used for? 99% of office work is emails, spreadsheets and writing letters. Sure, some printers don't work too well with Linux, so you might have to stick with XP to safeguard some old hardware, but with £5.5m we could have paid for some open drivers for all the

knackered old printers that are no doubt being used by councils up and down the land
3/ What on earth are we doing paying for Windows in the first place? Every year we see to hear a new pledge from somebody or other about how we've moving across to open data. How can this be true if we're locked into .doc format, or even worse, .docx?

James Taggart, Dundee

Graham says: James, we feel your frustration and share your bafflement. The good news from a long-term point of view is that if the city of Munich managed to switch to Linux, then we might one day too. We'll be writing to our MPs to kindly ask them to spend our taxes more wisely.



The Palace of Westminster, where the Chanel No5 in the executive washrooms never stops flowing.

PIE TIME

I agree that jEdit isn't pretty and deserves to finish last on that count alone [in our Group Test of text editors, in LV002], but it is an extremely powerful cross-platform editor. I installed it on my Linux Mint box, though in the past I've run it on both Linux and Windows machines just by unpacking the portable archive and directly running the appropriate executable file – no installation required.

In trying to keep an open mind, I tried Kate. Yes, it's good looking, but within two minutes, I hit a roadblock editing a LaTeX book I'm working on: it can only handle folding at `\begin... \end` blocks. With jEdit I have the option of enabling built-in folding on indent levels or explicit folding using `{{{...}}}` markers, or folding using the Sidekick plugin, which recognises `\chapter{}`, `\section{}`, `\subsection{}`, etc.

Why am I using jEdit instead of Texmaker? Texmaker doesn't appear able to do hard line wrapping on long lines of cut-n-pasted text, so I'm using jEdit to do that. I expect I could do everything I need with jEdit, but I haven't had the time or inclination to explore the options as I have a project to finish. So I'm using both Texmaker and jEdit on the same files. jEdit lets me know when the file has been changed on disk, whereas I must remember to reload it under Texmaker – yes, it's a mess, an expedient mess.

On my Mint box, jEdit seems to run quite happily with OpenJDK. However, sometimes the display doesn't refresh properly and needs to be forced.

I've done a lot of HTML with jEdit, of which this is an example: <http://oklanature.com/docentnews/DNL-201403.html>
Andrew Shead

Mike says: And this is the real reason free software is so good: we can choose what we like, and use what we want to create what we want. Jedit is still ugly as sin, mind.

HEARTBLEED

Forgive my presumption, but isn't it obvious that I'm going to scan the QR code on LV003 page 7?! Surely any self-respecting pot-o-gold special-prize hunter is going to go about scanning random QR codes as quickly as possible, in pre-release PDFs no less, to discover that ever-elusive secret hidden present... or maybe that's just me? Imagine my excitement when I find out that you guys love me! Especially as I love you all too.

Ooh, on that same page there's an alert to a GnuTLS bug? so not only did we have heartbleed this month in OpenSSL, but GnuTLS was also broken to such an extent that connections are vulnerable to man in the middle attacks! No wonder OpenBSD has forked OpenSSL to create LibreSSL in an attempt to improve the playing field for secured communications in *ix systems...

Related to Heartbleed I received an email from www.homeswapper.co.uk, who I signed up to many moons ago and then promptly ignored/forgot. The email in question says that Heartbleed "only affects Nginx and Apache web-servers running OpenSSL" and that their system runs neither of those two (they're actually hosted on Windows/IIS+ASP servers from what I can tell based on browsing around) and therefore they are unaffected.

Because of this lack of vulnerability, they claim I don't

3 QR codes considered for kernel crash messages

Kernel panic text isn't useful to many people, but if non-technical users could send a QR code containing the information to the relevant developers, that could help everyone.



need to change my password on their system.

What they neglected to mention is that they are not actually running any SSL-protected parts of their site! No, in fact all pages and forms, including privileged information about a visitor's housing arrangements along with passwords etc. are transmitted in-the-clear.

I have emailed them using their contact-us form to request that they follow-up their email announcing Heartbleed-invulnerability with another email announcing their "everyone and their dog, especially MITMs, vulnerability".

I have yet to hear back.

Daniel Llewellyn

Graham says: Heartbleed has the bejeesus out of a lot of people, but there are things that we can do to protect ourselves. As users, we can use a different password for each site we visit. And as administrators, we can follow the steps in Mike's tutorial on p86.

GAMES

May I offer a vote of thanks for your games coverage. Those of us who remember SuperTux as the highlight of gaming on Linux are revelling in the number of quality games on Linux today.

Dave Evans, Pontypridd

Graham says: You're right there. We're hoping that the Steam box in particular will tempt people away from Windows and into the fold.



Linux has finally come of age as a gaming platform. Praise be!

LUGS ON TOUR

SAPO Codebits

Josette Garcia has been in Portugal. It's a tough life sometimes...

For the seventh time I went to sunny Portugal to attend SAPO Codebits. I am always surprised by the creativity of the organizers. So what was in store for attendees this year?

There were 64 talks overall, and while there were talks on very diverse subjects, one could find unofficial tracks of 4–6 talks on one subject (gaming, security, hardware, etc); there were also two panels: one on game development and another one on Portuguese Makers, at the end of which the organisers announced the first Portuguese Maker Faire, to be held in September.

Babbage rides again!

Among the speakers were Christian Heilmann (Mozilla Foundation) and John Graham-Cumming (who is currently involved in a project to build Charles Babbage's Analytical Engine – see plan28.org), along with talks on building robots, cryptocurrencies, web design, 5G, creating an OS, etc. There were also two sessions of Lightning Talks, from freelancing to building arcade machines, via subjects as diverse as Tor, pair programming, and even

a 3D scanner (yes, a 3D scanner, the one thing that 3D printers are missing).

The core of Codebits is a 48-hour competition; people assemble in groups and hack together an original idea in just 48 hours; throughout the event, these groups are interviewed by the jury, which selects the best projects for the final stage presentation on the last day.

This year the winning project was an open source Knee Lock for polio patients so that they can walk safely without the typical metallic gears that were once used; the project was even featured on Wired (www.wired.co.uk/news/archive/2014-04/25/codebits-nelo); the project used a lot of things that were present at the event: 3D printers, Arduinos, gyroscopes, etc. Other projects delved into the realms of biosignals, VOIP solutions, stock markets, automatic localisation for product managers, a multiplayer version of the 2048 game and, of course, a version of Flappy Bird hacked together with a Kinect that had the players jumping up and down in real life!

There were also about a dozen satellite activities at this year's



For three days, 24 hours a day, 900 geeks were immersed in dozens of activities in the MEO Arena in Lisbon.

Codebits; these included a three-hour CTF security competition, a very funny geek quiz show, a zip-line crossing the venue, and even a competition of Nuclear Chili made with Bhut Jolokia, which is one of hottest chili peppers in the world (rated at more than 1 million Scoville heat units, roughly 400 times hotter than Tabasco sauce).

All of this put together, along with ~1,000 free meals every lunch and dinner and snacks and beverages available around the clock make Codebits an extraordinary event.

- The website: <https://codebits.eu>
- The event calendar: <https://codebits.eu/s/calendar>
- Some videos of the event <https://codebits.eu/s/page/videos>.



Codebits 2014 featured panels on games development and Maker culture in, leading to the announcement of Portugal's first Maker Faire.

TELL US ABOUT YOUR LUG!

Chances are that you are already a member of a Linux User Group (LUG). LUGs are all over the world and each one has its own unique selling point, which draws its members to meet and discuss their favourite topic. We want to know more about your LUG or hackerspace, so please write to us at lugs@linuxvoice.com and we might send one of our roving reporters to your next LUG meeting

Lincolnshire LUG

Dave Rice has news of a new user group.

Lincolnshire LUG is a new, virtual, online, user group and is probably the first of its type!

Lincolnshire is the second largest county in the UK covering nearly 7,000 square km, with over 1 million inhabitants. It has a wide variety of people living there; from farmers to large agricultural industry, metropolitan areas to tiny hamlets and, of course, it's the home of the Royal Air Force. With such a diverse economy the county can benefit in so many ways with a larger understanding of Free and Open Source Software.

The LincsLUG was initially set up many years ago to cater for the needs of the county's Linux users. It struggled, however, to become a widely used asset mainly because its members were all located large distances from each other and real meetings were difficult to organise. Unfortunately, the LUG fell into disuse and membership faded. Linux users in the City of Lincoln itself set up the superb Lincoln LUG, which is going strong; but the rest of the county, where Lincoln is too far

away to reach, were left without a LUG to call their own.

After a year or two, Dave Rice and Iain Baker, members of the original LincsLUG had a thought that because of the distances involved it might be worth a try re-launching the LincsLUG but as an online entity, enabling Linux users in all parts of the county to get together using the internet to discuss their favourite OS and the issues surrounding it.

Here there and everywhere

The new LincsLUG website is the focal point of the group and is in the process of being put together using tools that encourage online cooperation and enabling users to communicate and have meetings. We currently have a forum and an integrated Facebook page. There is an aspiration to have an embedded IRC channel and collaborative tools to enable common file storage and remote desktop help and support.

We still aim to have real meetings, but they will be less frequent and hopefully in conjunction with existing LUGs in the area. With the central point



Lincolnshire's geographical spread makes an online LUG a sensible option.

being the website, it will, hopefully, become a source of information for home users and businesses to find information about Linux and how it can help them as well as find a support network they can access at no cost.

Lincolnshire is a beneficiary of the rural broadband initiative – the whole county should have super fast broadband by 2016, and many local businesses are becoming more reliant on the internet as a means of doing business. So getting more Linux users will help keep business costs down and productivity up. With a successful LUG as an advocate to business and providing a support network Lincolnshire may well become a huge user of FOSS systems.

If you live in Lincolnshire, or just want to help us in our endeavours please visit <http://lincslug.org.uk> and join us.

OpenSUSEConf 2014

Free Software makes it to King's Landing.

April marked the latest OpenSUSE conference, held on the glittering shores of the Adriatic at the campus of the University of Dubrovnik in Croatia. Free Software can be a lonely furrow to plough, so it's essential for projects to meet up regularly.

OpenSUSEConf 2014 comprised four tracks: Business, which concerned itself mostly with implementation of OpenStack and OpenSUSE on ARM (we saw a sneak preview of this at FOSDEM earlier in the year); End User, with sessions covering the Jolla smartphone OS, among other

things; Technology development, and the much under-appreciated Community and Project thread. This included a look at how OpenSUSE can market itself more effectively; so we'll venture our opinion here and say: more SUSE beer please!

Video stars

The conference included 50 sessions, so even if you were there you probably missed something – but you've not missed out, thanks to the OpenSUSE YouTube channel, where you can see videos of the event's talks.



Once home to a medieval trading republic to rival Venice, Dubrovnik is now best known as a filming location in a TV show.



Ultimate Edition

Do you want one with everything?

The aptly-named Ultimate Edition distro is loaded with tons of Free and proprietary software. The latest version is based on Ubuntu 14.04 and uses the Mate desktop with custom theme, icons and pointers.

The distro has multiple apps for accomplishing the same task. In some cases, the multiple applications cater to users with different experience level – for example, there's Google Chrome for regular desktops and Elinks for advanced users. Users with ninja-level skills can use ImageMagick instead of firing up Gimp to tweak images. There's also LibreOffice, Blender, multiple screen recorders, VLC media player, XBMC media centre, the

Avidemux video editor and Blender for 3D modelling. For package management, the distro bundles the Ubuntu Software Centre along with the Synaptic package manager.

Script customisation

The most unusual aspect of the distro is its collection of custom scripts for performing certain desktop-oriented tasks. The scripts can all be accessed from the right-click context-menu, and you can use them to quickly shuttle files to another pre-defined location, convert images to different formats, mount ISO images, search for files in the current folder, and browse the current folder as the root user.



The latest version of Ultimate Edition is built from a debootstrapped Ubuntu 14.04 release.

PeppermintOS

If you have dumped offline apps and can work with the current staple of online apps, PeppermintOS is a wonderful alternative to ChromeOS. It's fast and is equipped with the necessary tools to get the job done.

The current stable version is based on Ubuntu 13.04 but the next version, Peppermint Five will be based on Ubuntu 14.04. "We base each version of Peppermint on the April release of Ubuntu. While we don't follow a strict release schedule, new versions of Peppermint can usually be expected each May or June", explains PeppermintOS's lead-developer Kendall Weaver.

The distro uses the Chromium web browser equipped to handle Flash content. Multimedia duties are handled by Gnome MPlayer for video

and the featherweight Guayadeque music player for MP3s.

Another lightweight app is the Mirage image viewer. The distro also includes several online tools such as Editor by pixlr.com and a bunch of games. There's also PeppermintOS's home-brewed app called Ice, which is a simple app that can turn web sites into web apps and roll them into the menu.

For package management the distro relies on Mint's Software Manager. Weaver and his team prefer the interface and user experience of Mint's software manager as compared with Ubuntu's. Weaver would also consider Lubuntu's software centre if it weren't for the ratings, reviews, and ease of categorisation of Mint's software manager.

PinguyOS

PinguyOS is different from the others in many ways. For starters, it tags its six-monthly releases based on the current Ubuntu release as Beta releases. Only PinguyOS releases based on an Ubuntu LTS are considered stable.

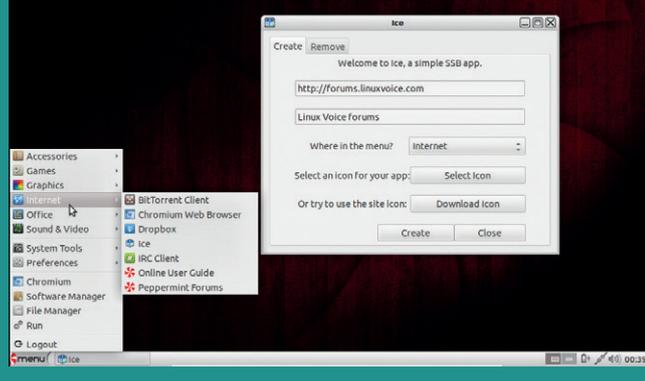
PinguyOS is designed to appeal to users who are new to Linux. The developer has modified the default Gnome installation with some extensions. Thanks to these you can place icons and folders on the desktop, and can minimise windows. The distro has also swapped Gnome's Activities Overview with the GnoMenu,

which combines features of both the modern and the traditional application menu. PinguyOS also includes Ubuntu Tweak and the Gnome Tweak tool to help you configure many different aspects of the desktop.

Game management

Gamers will enjoy the inclusion of the Steam client, PlayOnLinux front-end for Wine and the DJL game manager, with which you can download and install several free, freeware and shareware games. The distro also includes several proprietary applications, such as Dropbox, Skype and TeamViewer.

PeppermintOS uses a Site Specific Browser (SSB) to run applications on your desktop.



The latest release of PinguyOS based on Ubuntu 14.04 will feature the Plex media server and make sure that Netflix works out of the box.



What's new in Ubuntu 14.04

A whistle-stop tour of the latest release.

As far as the official Ubuntu versions go, the Long Term Support releases aren't all that exciting. They introduce no new features and fuffle no feathers, and are generally a bit boring; but that's a good thing, because stability is a feature. The goal of the LTS releases is to provide users with a steady base that they can continue to use for a long time.

Ubuntu 14.04 LTS (codename: Trusty Tahr) is no different. Released on schedule two years after the last LTS release, most of the changes in this release are fairly conservative in nature, and mostly hidden out of view embedded deep with the code. As with every release, all the default applications and core components have been updated to their latest stable versions. There's Firefox 28, Thunderbird 24.4.0, LibreOffice 4.2.3, Rhythmbox 3.0.1, Nautilus 3.10.1, and the distro is powered by Linux kernel 3.13.

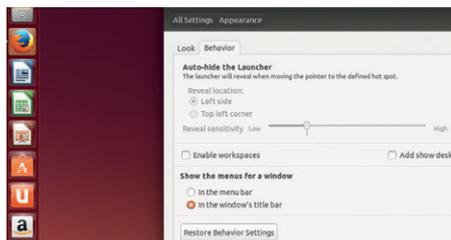
At first glance, the Unity desktop in Trusty doesn't look any different from the one in the previous Saucy Salamander release. However there are minor visual tweaks that long-time users will notice (and appreciate) as soon as they start using the distro.

For starters, instead of the Compiz decorations, Unity has switched to GTK 3 CSS-themed window decorations. You'll

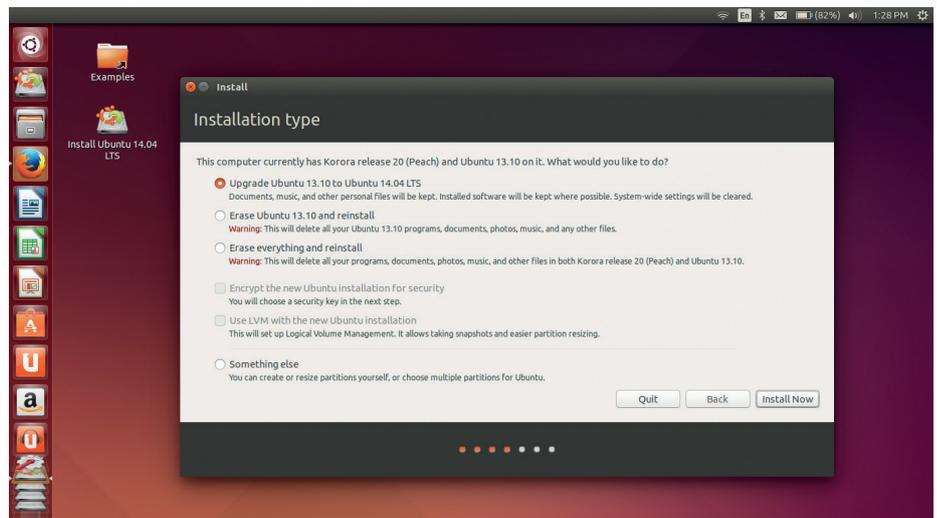
“One of the most prominent additions to the latest Ubuntu LTS version is the ability to turn off the global menu bar at the top of the screen.”

notice that windows in Unity aren't bounded by a one-pixel black line and are now completely borderless. Another benefit of the move from Compiz to GTK 3 is that the window corners are now antialiased.

Also you can now resize windows in real time. Until now, resizing windows in Unity



Users can now turn off global menus. Hurrah!



One of the strongest points of Ubuntu is its easy-to-use installer.

would display a frame with the new window size, and the change would only happen after you were done resizing the window.

More usable Unity

The release also fixes a couple of long-standing usability issues. You can now right-click on the global menu even when a window is maximised. This displays the same menu that's available when you right-click on the titlebar of an unmaximised window. Also, you can now tweak the

behaviour of windows when you click on their icon in the Launcher. You can, for example, set this action to minimise an open window. However, this feature is implemented as an “unsupported” extra and you'll need to grab the CCSM app from the Software Centre to enable it.



Ubuntu ships with more lenses than ever before.

One of the most prominent additions in Ubuntu 14.04 LTS is the ability to turn off the global menu bar at the top of the screen. It's been a mainstay of the Unity desktop since its introduction in Ubuntu 11.04, and despite several changes over the years, hasn't managed to impress all users. In the latest release there is finally the option for locally integrated menus (LIM) in an app's title bar. By default, menus appear in the top panel. To move them closer to the app, head to the Behaviour section under the Appearance tab in the System Settings panel and click on the Local Menu option.

Once enabled, the app menus are embedded directly in the window decoration and continue to save the screen real estate. When the menus are wider than the available space, the extra items are tucked underneath a small drop-menu at the corner of the window. The new menus don't have any impact on the usability of the windows – you can still move them just as before.

When you step away from the computer, Ubuntu will trigger the new Unity lock screen, and the Unity Spread view (which is triggered when you click on an app's icon in the Launcher with open windows) now lets you filter the windows by their titles.

On a sombre note, Ubuntu 14.04 LTS will also be the first release to ship without the Ubuntu One cloud storage service. Canonical pulled the plug on the online storage service along with the Ubuntu One Music service, a couple of weeks before the final release.

LinuxLite

LinuxLite is designed for people who are new to Linux, and is ideal for under-powered machines. The distro is intuitive to use and is based on Ubuntu LTS releases. To acclimatise users, LinuxLite runs a heavily modified version of the XFCE desktop environment.

Despite its name, LinuxLite ships with lots of software – including Thunderbird, Gimp and Firefox with the Flash plugin – and in the Games menu you get links to buy Humble Bundle games, along with an entry that displays essential information about the hardware requirements

to run Steam. LinuxLite is one of the few distros that includes a Help Manual complete with an illustrated installation and post-installation user guide.

While the distro uses the Synaptic Package Manager, it also includes menu entries to quickly install frequently used apps and packages. You can, for example, install individual pieces of software such as the Deluge torrent client, the Steam client and VirtualBox, as well as bundles such as the games pack, IA-32 libs, multimedia codecs and more.



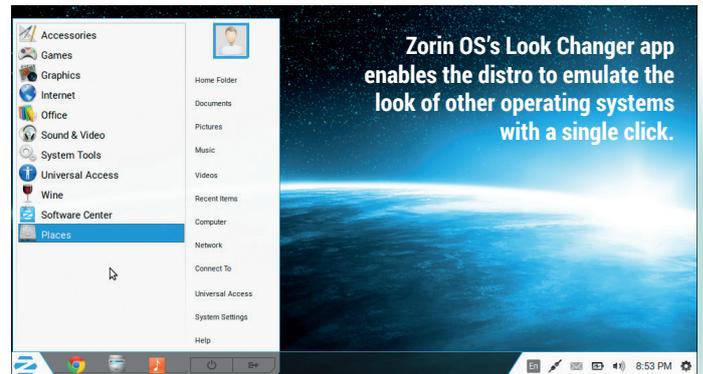
Zorin OS

Zorin OS isn't the first distro that's aimed at easing Windows users into a Linux distro. Lindows and Linspire are probably the best-known examples of distros that have tried (and failed) to put a familiar face on top of a Linux distro.

But Zorin OS, with its Gnome desktop modified to resemble Windows 7, does an admirable job of lowering the learning curve for new users. "Zorin OS was built with the goal of being

the best desktop operating system for Linux newcomers, with emphasis on Windows users making the switch to Linux," explains Zorin's lead-developer Artyom Zorin.

Zorin OS is based on the latest Ubuntu release. The idea, Artyom explains, is to "build on the greatest that Linux has to offer and package it into a system that anyone can use without learning anything new thanks to its familiar interface."



ElementaryOS

But of course!

If you think Linux distros can't be pretty, check our ElementaryOS. The distro uses custom icons, custom themes and custom apps to produce a beautiful Gnome-based desktop. The distro includes apps like the Shotwell photo manager, Empathy IM, Totem movie player, and Gparted and quite a few custom apps including the Scratch text editor and the Geary mail app.

The first Elementary app was its file browser, which was created because "we simply weren't happy with the interface and features provided by the available browsers", explains the distro's lead developer Daniel Foré. "The same story played out for our Music player, Calendar, and the prototype mail app we started work on before Yorba came out with Geary."

According to Foré, the customisation also extends behind the scenes. The project created a service called Contractor because the team wanted the apps to be interconnected and extendable without requiring developers to know which other apps were installed on a system, since so

many free desktops are highly customised setups. Another example he highlights is that of the Granite library that hosts commonly used complex widgets that hadn't made their way into GTK, like the TabBar, search bar, popovers, etc. This ensured consistency across apps and centralised the development of UI elements.

Dedication to beauty

Elementary's Gala window manager is built on LibMutter, which offers the developers more control and better performance than Compiz. They wrote the stylish top panel they wanted to be in the same ecosystem as the Ubuntu app indicators without using Unity. "In short" says Foré, "we end up writing apps because we're just not happy with the quality or architecture of existing solutions".

One surprising element in the distro is the use of the Midori web browser instead of a more mainstream browser like Firefox or Chromium. "We have a strict dedication to providing a consistent, native experience in elementary", explains Foré. "We only ship



ElementaryOS will appeal to new as well as regular Linux desktop users.

GTK+ apps, which is why we don't ship apps like LibreOffice, Firefox, Thunderbird, etc." Package management is handled by Ubuntu's Software Centre. The distro uses Ubuntu's repos, but also has a PPA of its own for providing updates for the custom apps, as well as for apps and libraries that the developers think are important. "In contrast with Ubuntu, we continue to make both bugfix and feature releases of apps throughout the life cycle of the OS release."



Centrych

Centrych Get the best of both worlds.

Here's a distro that lets you run KDE apps but has the footprint of an Xfce distro. Centrych OS lets you run both Qt and GTK apps but without the usual bloat that accompanies these libraries.

Explaining his motivation for labouring on the distro, Centrych's developer Jack Radigan says: "I felt that there was an unmet need for an easy-to-use, single desktop designed to support applications written for either the Gnome/GTK and KDE/Qt environments. KDE can support both, but it's a much more complex environment and I wanted to provide a simpler, more accessible alternative."

"I'm targeting a mindset rather than any specific audience", says Radigan. "There's quite a few capable productivity applications today, some written for Gnome, some for KDE. Those who prefer using the best from both worlds should give Centrych a try."

Radigan started the project with the intention of creating a lightweight GTK-based distro that could run a few Qt apps without running into dependency issues.

The distro uses the Xfce desktop environment with two distinct profiles. The default profile has the look of KDE along with the Oxygen theme, while the other provides the GTK look of Xubuntu together with the Greybird theme.

Centrych has also swapped out the default Xfce menu with the Whisker menu. Radigan also hopes that the distro appeals to XP users who'd rather not toss their hardware investment after Microsoft halts security support.

KDE and Gnome

Both profiles give you access to the best of Qt and GTK apps. There's the Clementine music player and Kopete IM along with Gimp, Gnome System Monitor, VLC media player, LibreOffice and Firefox. For package management you get both Synaptic and the Ubuntu Software Centre.

The distro is based on the Ubuntu 12.04 LTS release, but Radigan makes sure that users get the latest versions of certain "high-profile" apps such as Gimp or LibreOffice.



With Centrych you can use popular KDE and Gnome apps on machines that can't power the full desktop environments.

He also provides updates for the nearly 300 packages that have been modified for Centrych. For everything else, he relies on updates from upstream Ubuntu.

Radigan is currently working on the next version based on the Ubuntu 14.04 LTS release. "This release is about refining what's currently in place, since it's only been about a half year since the first release came out."

Bodhi

If you like to build your distro by hand, but don't want to go the extreme of Arch or Gentoo, then Bodhi Linux is for you. It ships with a small number of apps, and you can add more using the distro's web-based software installation tool called AppCenter. Using the tool you can also download packages on any machine and then bring them over to Bodhi for installation.

Bodhi uses the current Ubuntu LTS release as a base and adds

the Enlightenment desktop environment, which manages to be both responsive and beautiful at the same time. According to Bodhi's Jeff Hoogland, Enlightenment "has resource usage that is comparable to LXDE/Openbox while being far easier to customise... By building on top of Ubuntu LTS releases Bodhi allows users to have a rock solid base operating system that doesn't need a reinstall or a massive upgrade every six months."



Bodhi Linux is perhaps the best integrated Enlightenment distro and ships with a bunch of profiles optimised for different types of hardware.

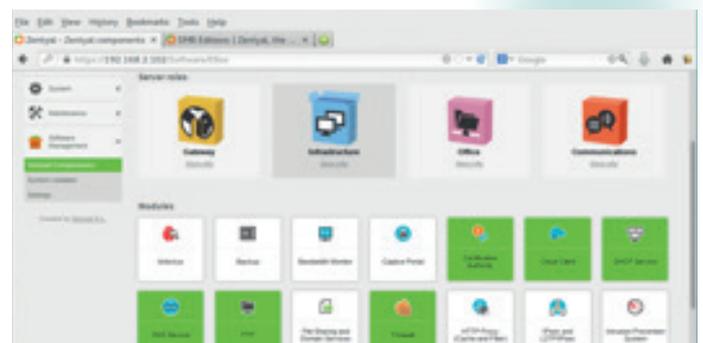
Zentyal

Zentyal has all the components you need to run a gateway server, an office server and a communication server. It's got the Apache web server, OpenLDAP directory server, Bind DNS server, Jabbered2 IM Server, Zarafa groupware, Asterisk VoIP and DansGuardian for content-control management and a lot more.

You can monitor and control the various components of the server using the distros custom

management tools. Zentyal goes the extra mile to help you configure the different servers and services without mucking about with configuration files.

Once the distro is installed, it will start a configuration wizard to help you setup the server. From here you can install individual server packages, or modules in Zentyal's parlance, or select predefined groups such as Gateway, Infrastructure, Office, and Communications.



You can also install Zentyal on top of an Ubuntu Server installation by adding and pulling packages from Zentyal's repository.

Official Ubuntu spins

Unity ain't for everyone.

Choice is the hallmark of open source software. Even though it focuses its efforts on the home-brewed Unity interface, Canonical supports other desktop environments as well. In addition to the main Ubuntu distro, the project has several official spins that swap out the Unity desktop with other popular desktop environments. However, the developers of these official spins go the extra mile and introduce other changes as well to suit their target audience.

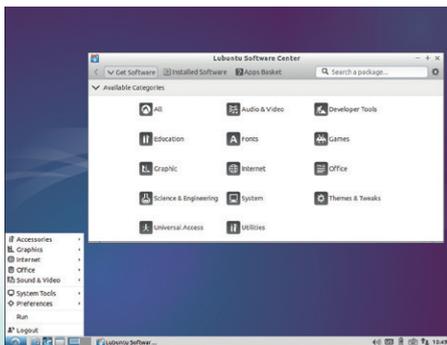
Ubuntu Gnome

The Gnome 3-based spin is the latest addition to the Ubuntu stable of official derivatives. It includes popular apps such as LibreOffice, Evolution, Shotwell, Rhythmbox and Totem. However, instead of shipping Gnome's default Epiphany web browser (now rechristened simply as Web) Ubuntu Gnome includes Firefox. The distro is also missing some other Gnome tools like the Boxes virtualisation app. To help you customise its Gnome desktop, the distro includes the Gnome Tweak Tool app.

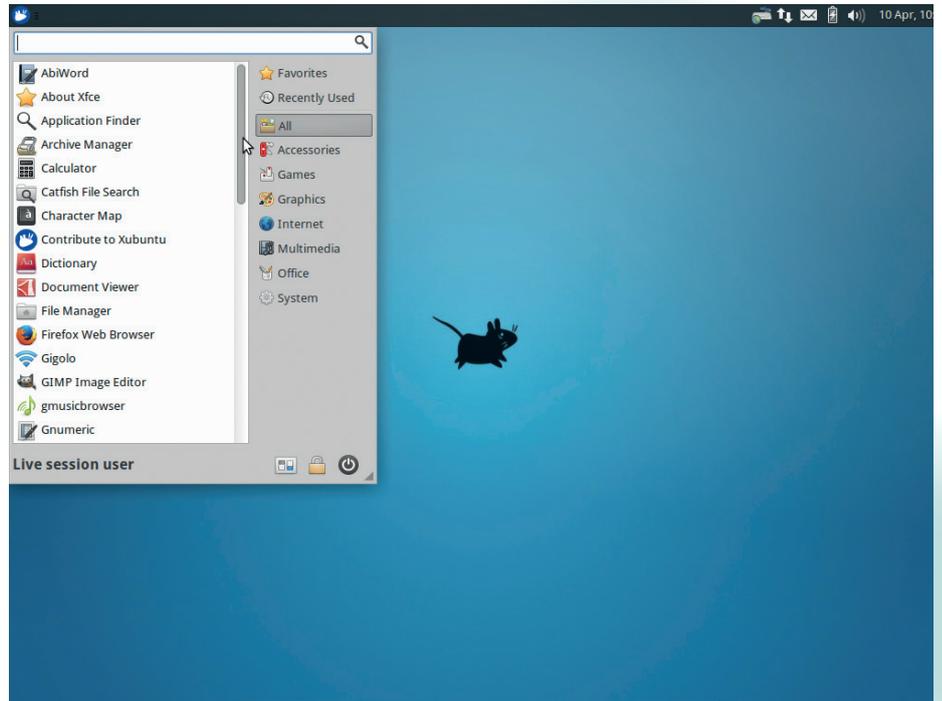
Ubuntu Gnome 14.04 is the first LTS release for the project and will be supported for three years as opposed to the five years for Kubuntu. This latest release will ship with Gnome 3.10, which was released back in September. If you're interested in the latest Gnome 3.12 release, you can upgrade to it using the Ubuntu Gnome team's PPA.

Kubuntu

This is the oldest and one of the most popular spins of Ubuntu, offering the KDE desktop. The distro includes KDE apps including Kmail, K3b, Amarok and Dragon Player in Kubuntu as well as their popular apps such as Firefox. It skips the KDE office



Lubuntu is the lightest official Ubuntu spin that performs well on older hardware.



Over the years Xubuntu has become the go-to distro for Gnome 3 and Unity refugees.

Calligra Office suite and instead ships the more popular LibreOffice.

The latest release, Kubuntu 14.04, is based on KDE 4.13. One of the many improvements in this release is that the Additional Drivers app has been replaced with the all new Driver Manager, which enables you to install and select the drivers you want to use. The built-in recommendation software will notify you when better drivers are available for your hardware, including proprietary ones.

Lubuntu

If you want Ubuntu goodness on an under-powered computer, there's Lubuntu, which is based on the LXDE desktop environment. The distro also replaces heavyweight apps with lighter alternatives. This is why Lubuntu includes AbiWord and Gnumeric instead of LibreOffice.

Lubuntu ships with some GTK apps such as the Evince document viewer, Archive Manager and mtPaint image editor along with featherweight apps that go with its LXDE desktop, such as the Leafpad text editor and PCManFM file manager.

Lubuntu also has the Sylpheed email client as well as the Firefox web browser. It's also got Audacious and Gnome MPlayer,

which can play most popular formats including MP3s, AVIs and MP4s.

The developers of Lubuntu have created a lightweight version of the Ubuntu Software Centre called the Lubuntu Software Centre. The tool also has an Expert mode for installing individual libraries. Lubuntu 14.04 will be the distro's first Long Term Support release and will be supported for three years.

Xubuntu

Another spin that's designed for relatively older hardware is the Xfce-based Xubuntu. The 14.04 release of Xubuntu uses the Xfce project's latest desktop manager, Xfdesktop 4.11. This release will use the Mugshot user account profile editor and the Light-locker screen lock instead of Xscreensaver.

The distro will also switch to the Whisker menu launcher. Not surprisingly, Xubuntu doesn't include a full-fledged office suite, but rather the AbiWord word processor and Gnumeric spreadsheet. Surprisingly though it includes Gimp and other popular apps such as Firefox, Thunderbird, Pidgin and Transmission. There are some lesser known apps as well, such as the gmusicbrowser jukebox, Parole media player, and Ristretto image viewer. The distro uses the Ubuntu Software Centre for package management.



Netrunner

The drop-in Ubuntu replacement.

Don't let its name fool you. Although the distro has tools to consume online information, Netrunner can be used productively offline as well. In fact, the distribution does a pretty nice job of bundling the new online apps that run within a web browser and need a connection to the internet along with traditional offline apps for regular heavy-duty desktop tasks.

Netrunner is based on KDE, but that doesn't mean it's just another Ubuntu-based distro that has bolted KDE in the place of Unity. Unlike Kubuntu, which ships with more or less a stock KDE release, Netrunner has exploited the desktop's customisability to the hilt and ships with a much more attractive and functional environment.

The developers put in considerable effort to chip away the rough edges of the default KDE environment and make the desktop appealing to new users. The Netrunner desktop follows the standard desktop metaphor with a few refinements. Instead of switching to a revolutionary new desktop, Netrunner has introduced subtle refinements to the existing elements.

The distro uses a modified version of the Homerun launcher, known as the Homerun Kicker. Unlike the full-screen Homerun launcher, Homerun Kicker resembles a traditional launcher menu and is designed to be operated by the mouse, keyboard and even touch. Along with the traditional cascading popup menus, Homerun Kicker also includes a sidebar strip where you can pin your favourite apps. There's also a search box that mines several different sources of data.

Another useful feature is the side panel. This is a normal Plasma Panel that includes the Veromix widget (to control sound



Netrunner ships with a beautiful rendition of the KDE desktop.



The distro has all the usual sources for dispensing help and also publishes an online magazine

volume), an analog clock and the media player, which can playback any media even when hidden. You can bring up the side panel with a pre-defined gesture. Netrunner also includes the Easystroke tool, with which you can define your own custom actions and gestures to activate them.

Full-service distro

You can use Netrunner straight out of the box. The distro is chock full of apps, both Free Software and proprietary. There's Skype, Dropbox, Thunderbird for email, and the Firefox web browser is equipped with plugins to play Flash, Java as well as the AdblockPlus and DownloadHelper add-ons.

The distro also ships with codecs to play all sorts of audio and video content and includes the Clementine music player as well as VLC. Along with the LibreOffice suite there's also Calligra Flow. Netrunner also includes the Wine compatibility software for running Windows apps, and the Wine Tricks utility, which can download and install several popular proprietary apps. For the gamers out there, Netrunner includes the Steam installer.

One of the best cloud-oriented features of the distro is the integration of its Runners-ID cloud storage service. The service offers

2GB of free space that you can use to store data, pictures, contacts, calendars, and stream music via its Android app. Netrunner's Webaccounts app helps you integrate Runners-ID to the respective native KDE app. In addition to Runners-ID, you can also use the Webaccounts app to connect to and pull in data from other online services such as Facebook, Google, and OwnCloud.

If you need to flesh out the default software in Netrunner, you can do so using its three package managers. Experienced users can use the old school Synaptic Package Manager and new users can use Muon Discover or the web-based JackNJoe service to install packages.

A magazine about Linux!

Help on the distro is dispensed via online forum boards on its website, which also includes quite a few demonstration videos and screencasts. The developers behind the distro also publish an online magazine called Netrunner-Mag.

The Netrunner distro is perhaps one of the most underappreciated ones, lying at the #59 on **Distrowatch.com**. All things considered, Netrunner is a polished KDE offering and offers one of the smoothest user experiences.

Trisquel

A typical Ubuntu installation has several proprietary bits and blobs, mostly to support the latest hardware – particularly graphics cards. Not surprisingly, the Free Software Foundation has guidelines that define a free distro. There are a couple of distros that adhere to those strict guidelines at the cost of usability. Trisquel, however, is a wonderful exception.

Trisquel is available in several editions. The main release is a 700MB installable CD, which uses the Gnome desktop. There's also a 500MB mini edition designed for older hardware, which uses the

LXDE desktop. The distro is based on the Ubuntu 12.04 LTS release. It uses Gnome 3.4.2 and runs in the fallback mode to offer the traditional desktop experience.

Trisquel has all the apps that you'll find in other distros, such as Evolution, Pidgin, Gwibber, Transmission, LibreOffice, Gimp, etc. The one oddity is the web browser: Trisquel ships with Abrowser, which is the unbranded version of the Firefox web browser. The developers have also modified the browser to fetch add-ons from Trisquel's own repository instead of Mozilla's, which houses some non-free add-ons.



UberStudent

UberStudent uses the Xfce desktop and includes a wonderfully curated combination of hosted apps and online ones. For example, the Books menu points to the only installed app, which is the FBReader ebook reader. The other entries in the menu are to various online resources where you can find ebooks, such as

Open Library. Another category is Personal Finances, which points to the online Mint Personal Financial Manager app and also helps you keep costs in check by pointing to websites that let you buy used books, rent new ones, and even search for deals on Air fares. The distro also has a handful of games and other general-purpose utilities.



Emmabuntüs

The kitchen sink flavour.

Emmabuntüs was designed to ship with reconditioned computers assembled by humanitarian organisations from donated pieces of hardware, and owes its name to the French Emmaüs charitable movement. However, it'll fit snugly on any desktop.

The current version of the distro is based on Xubuntu 12.04.4. Since many of the intended users of the distro wouldn't have access to the internet, the distro is overflowing with apps. Emmabuntüs includes virtually every popular open source app and also includes all sorts of proprietary codecs and apps. No wonder then that it is available as a huge 3.3GB ISO image.

In addition to regular desktop apps such as LibreOffice, Gimp, VLC media player, OpenShot video editor, RecordMyDesktop screencasting app, there are apps for power users such as BootRepair and VirtualBox as well as Wine, WineTricks and PlayOnLinux. The distro's two web browsers, Firefox and Chromium, ship with lots of plugins to block ads and prevent phishing attacks. This helps

make the distro suitable for young users who'll also enjoy the plethora of educational apps and games, including the OOo4Kids office suite for children.

All about choice

The distro also does a nice job of educating its users about the proprietary components before installing them. Emmabuntüs bundles the proprietary apps but doesn't install them by default. The Cairo Dock is one of the highlights of the distro, according to its lead developer Patrick. The distro ships with three variants of the dock and you can switch between them from within the distro itself. The default "Simple" version is designed for new Linux users, a "Kids" version is meant for children and a "Full" version for experienced Linux users.

According to Patrick, the team will maintain the current version, Emmabuntüs 2, for another year with release updates every three months. "During that year we will stabilise Emmabuntüs 3, which will be based



Emmabuntüs can easily install proprietary software without a connection to the internet.

on Xubuntu 14.04." Once the new version is out, the previous release will get updates every six months until April 2017.

Xfce desktop and the Cairo-Dock make the desktop very approachable to a far wider userbase than its intended audience – a bit like the way every Ubuntu spinoff in this feature has taken the fantastic features of Ubuntu to a wider audience than Ubuntu could have reached on its own. 

KANO

TO ENABLE?

What is the Kano project, and how does it enhance what the Raspberry Pi is already doing? Les Pounder finds out.

The Raspberry Pi has been with us for over two years, and in this time there have been many projects created using this board. Projects such as Pi Supply, MotorPiTX and PiGlow have aided many hackers in their quest to build something new with their Pi. And herein lies an issue that resides at the heart of the Raspberry Pi Foundation. The original goal of the Raspberry Pi and the Foundation itself, was to create a low-cost educational tool enabling children around the world to learn computer science, programming and electronics. Its success in this goal is steadily being realised by an ever-growing education team, and at the start of April 2014, the Foundation updated its website and included a series of tutorials, written for absolute beginners to follow. This update was also supported by a series of training sessions, called Picademy, for 24 teachers from across the UK. These sessions sought to spread the concept of using the Pi in education via a carefully curated lesson plan.

The Kano team have a similar goal to the Raspberry Pi Foundation: they wish to use the Pi in an

educational environment and enable children to learn more about computing.

However Kano approaches this issue in a different manner by creating an intuitive “computer lab” in a box, that is built around the Raspberry Pi. The design and production of the Kano products, such as the accessories, step by step book and stylised packaging

“Kano creates an intuitive computer lab in a box, that is built around the Raspberry Pi.”

has stood the project out in a sea of many derivatives but the Kano project has also polarised the Raspberry Pi community, with some seeing the project

as riding on the success of the Raspberry Pi, while others are looking forward to using the kit in their next project. Read on and decide for yourself...

History of the project

Just over a year and a half ago Alex Klein was completing his graduate studies in computer science at Cambridge University and working as a writer for Newsweek and The Daily Beast in the US. During this time he ran into Eben Upton. Eben and Alex got to know each other bit and they discussed how

essentially the Raspberry Pi had been introduced to meet the decline in students opting to study Computer Science, but that Eben had never expected it to sell more than 20,000 to 30,000 units. Eben also told Alex that the core social intent of the project was to get kids and beginners interested in coding again. Alex looked at the user base at that time and most of the users were around 44 years of age and male so the Raspberry Pi was, at that early stage, not reaching the target demographic that they wished for. Instead it was being adopted by users of a generation that remembered the BBC Micro, ZX80 etc. Other groups were also using the Raspberry Pi in all manner of maker-style projects, such as home automation and low power home server projects.

Democratisation

"We're living in a time where a small Linux board is cheaper than a textbook, you can get internet anywhere and you can learn almost anything for free online and yet I look around to my younger siblings and friends and realise that we have no idea what is going on inside our devices and these devices control almost every aspect of our lives," Alex later told us about his early inspiration.

Alex's older cousin Saul introduced him to Yonatan Raz-Fridman, who would become another Kano co-founder. They picked up some Raspberry Pis, and wanted to see how a real life kid would use the kit so they went to Alex's little cousin Micah and asked "What do you want to build with this?" to which he replied "A Computer". Alex and Yonatan tried to do that with him, but it was really quite difficult and Micah was quickly flummoxed. Alex and Yonatan asked themselves about what could be done about this, and Micah replied "Make this as simple to use as Lego".

That led the project to start writing a step by step book similar to those used when building a Lego



Alex Klein, computer scientist and Kano founder, proudly wields the Kano's speaker module.



model, something which is inherently intuitive for children to follow. In the book, Kano covers simple steps and projects such as how to connect your Raspberry Pi, how to run your first script and how to watch a video from YouTube. The book uses a simple vocabulary to ensure that any level of reader could follow the steps.

Testing the project

The Kano project began to make kits, comprising the book as well as the SD card with custom Kano OS, keyboard and all the cables to connect a Pi to a monitor. Once they were in a position to beta test they took early prototype kits to schools around the UK. Their goal was not to teach a lesson, but to see if their kit was intuitive enough for children to learn via discovery. Alex and his team would walk in and say to the kids "In the next hour you are going to build a computer, make games and make music. And guess what? I won't need to teach you how to do it, you are going to do it by yourself using these kits"

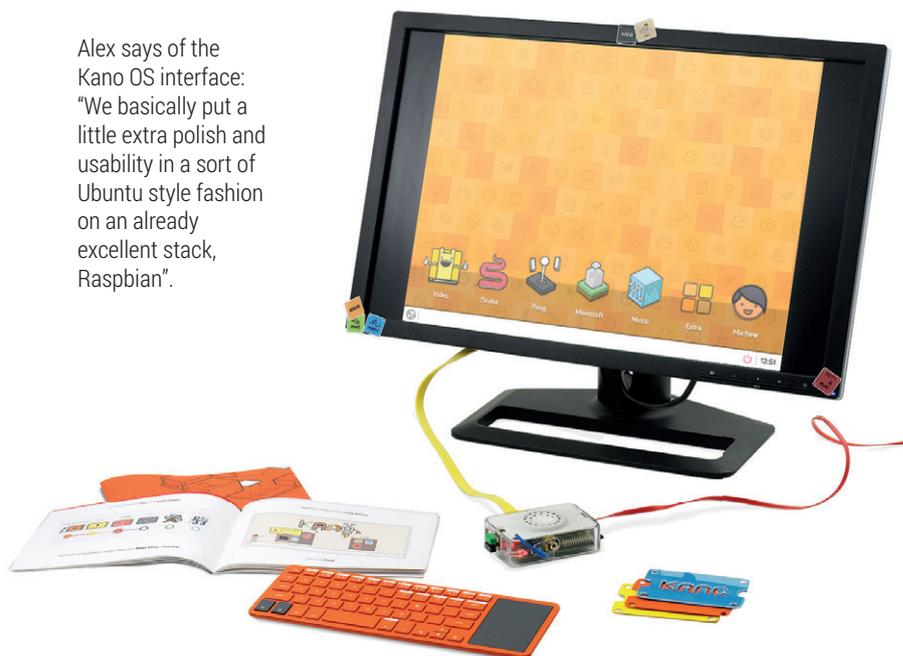
After the first workshop using these early prototypes, it was clear that it was a success and that children enjoyed using the kit. Alex happily recalls receiving feedback from one of the children "A child named Khalid who was nine years old said to Alex and the Kano team "Adults believe that we are incapable because we are young, but today we made a computer so that makes us Super Children"

Kano believe that this kind of learning will help children "to find a way to make open source and hardware hacking feel to them, not like a chore, but like a game." Alex went on to say: "We felt that it should be like crossing a Commodore 64 with a Nintendo 64 with the benefits that each brings."

From here Kano worked with industrial designers to create the physical aspects of the kit, like the speaker and modular case. Alex goes on to say "We had what we thought was a pretty sexy piece of kit, and we then took it to Kickstarter to help fund our growth. We wanted to make around 1,000 kits, which would cost us £95 to make. So we wanted to crowdfund the investment necessary to build this many kits."

On the last weekend of July 2014, 16,000 people in 87 countries will open up an end-to-end, Raspberry Pi-powered kit, with a Debian Linux brain and dozens of projects to get them started right away.

Alex says of the Kano OS interface: "We basically put a little extra polish and usability in a sort of Ubuntu style fashion on an already excellent stack, Raspbian".



On 19 November 2013 the Kano team had a launch party with friends and family. Alex talks about launching the Kickstarter "During the party we decided to press the button on our Kickstarter campaign and something wonderful happened. The demand was fantastic: we wanted \$100,000 and we managed to raise that in 19 hours. After 30 days we raised \$1.5M including support from Steve Wozniak, co-founder of Apple, and Yancey Strickler, the co-founder of Kickstarter." Alex then goes on to say "At this stage we were blown away, but knew that we had a

The Kano kit has 21 unique components, all of which are either designed top-to-bottom or reskinned.

responsibility to bring something new, fun and truly pedagogically sound to market."

Kano's operating system is based on the rock-solid base of the Raspbian team's hard work. Pretty much everything that you can do with Raspbian, you can do using Kano. Alex was keen to tell us more about the user interface.

Kano OS

"From a user perspective the interface is different to Raspbian. We have a beautiful apps layer with apps that are very much catered to beginners, but they are also a lot of fun to use. Kano also comes with a suite of user-friendly tools to make your first project a little bit more simple. The simplicity enables you to focus more on the code rather than the application."

Kano is also the first project to integrate a visual programming language with Minecraft, using the Minecraft API to enable the use of Python to create new content in the Minecraft world. The process is similar to the functionality of Scratch, where you drag code blocks to build your program. These blocks are then translated in to Python or JavaScript which is then output via the Minecraft API to the Minecraft world. Alex explains "So with this you are getting this really cool side-by-side view of what you are coding, be it a building or a volcano, and you can see the code that made it possible."

But Kano is not just a shiny new interface and some simple applications: the back-end of Kano OS has other improvements, Alex explains "Behind the scenes we've really increased the speed of Kano with a 50% increase in speed for the browser, GTK, Linpack and in the overall boot time, compared to stock Raspbian". Kano also simplifies the configuration across the operating system, from Wi-Fi, to the desktop manager, to keyboard and audio configuration. These simplifications are welcome to new users who just want to experiment with the platform or are unfamiliar with a Linux distribution.

Open source

These improvements are not exclusive to Kano, as the team intend to share their work with the community. As Alex explained about the team's plans, "We are an open source software company and our improvements will be pushed upstream to the Debian kernel. We'll also share our projects and educational content back upstream."

The team have already opened most of their GitHub repositories, and the full OS source is readily viewable in the image. During the course of our conversations with Alex it came across very clearly that Alex and his team are giving back to the Raspberry Pi community – despite what some have said.

Alex explained, "We're giving back with apps – a beautiful desktop manager, intuitive system settings, a smart package system, etc – that significantly develop the platform, as well as games and projects that are accessible and awesome."



Alex is also keen to promote open source to children “To help children engage with open source, it has to be presented to them in a playful, “Lego” kind of a way. There is a playful and an educational aspect to the project and we are trying to promote this kit to the mainstream under that pedagogical context.”

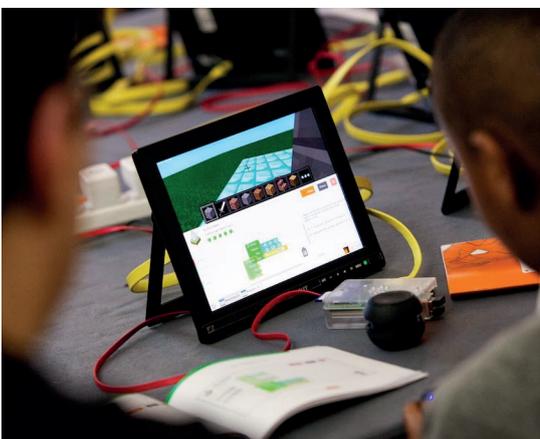
The future of the project

The Kano project is moving towards an investment round (where companies seek investment from venture capitalists), which would make Kano the first Raspberry Pi-based company to receive VC investment. It has recently put in a major order to Farnell for 18,000 Raspberry Pis, which makes Kano the largest purchaser of Raspberry Pi in the world.

At this time the Kano project is a singular kit, but the team have plans to expand the Kano brand via a series of hardware and software products. Alex explains: “We see Kano as a creation kit for many levels. The basic kit, which is building a basic computer, is what we have at this time and it comes with many hours of learning and enjoyment, but just like video gaming there will be expansion pack.”

He goes on to explain what the packs will consist of “These expansion packs will constitute both software and hardware components, such as a project to build a battery and LED to make your Kano portable, then the next expansion pack could be a screen to plug in and make your own tablet, and then a future expansion to use the camera with your tablet.” There has been a co-ordinated effort between Kano, industrial designers and Farnell to roll out these expansion packs in a timely fashion.

But Kano is not just about the technology and the packaging; there’s also an educational element to the project via the Kano Book and a companion guide. The book, initially written by Alex, is a simple step-by-step guide to using the Kano kit, as Alex elaborates “I started writing a step-by-step book similar to those used when building a Lego model. In the book we cover simple steps and projects such as how to connect up your Raspberry Pi, how to run your first script and how to watch a video from YouTube. We



The Minecraft API enables users to see features they have programmed in the game, in real time.

Kano and the Raspberry Pi community

In recent time there was a small issue in the Raspberry Pi community, when one of the Kano team, Alejandro Simón, claimed via anecdotal evidence that “It wasn’t friendly for the teachers,” he said. “They received [Raspberry Pi] kits and massive instruction books and they weren’t prepared for it – so they were gathering dust.” This statement caused some polarisation in the community, but when we spoke to Eben Upton, from the Raspberry Pi Foundation, he clarified the

status between Kano and the Foundation. “We support anyone (and there are quite a few examples, including our first-tier partners RS Components and Premier Farnell, and Foundation favourites like Pimoroni and ModMyPi) who bundles the Raspberry Pi with an education-focused kit of peripherals. To give him his due, Alex realised that Alejandro overstepped the mark in claiming, without evidence, that significant numbers of Raspberry Pis were ending up in drawers”

used simple vocabulary to ensure that any level of reader could follow the steps.” The book’s vocabulary is squarely aimed at children and is grouped into “Levels” with level 1 being the easiest task. These books are easy to use, and are intended to encourage learning via discovery.

Self-guided learning

Alex discussed this topic at great length, and it was clearly a passionate subject: “We have the kit, which is self contained, and we want the kit to be as easy to use as Lego not just for children, but for their parents and teachers. We want teachers and parents to be involved in the learning and not just spectate, so our documentation must reflect the goals of the projects and illustrate them clearly in a story-like way. So what we are going to be doing is a companion book, which is not necessarily a curriculum guide or a set of lesson plans but will be a support framework from which teachers can build their own lesson plans. An example of the support framework is in the main Kano book: Level 1 of Kano is to make a computer, which is all

about setting up your Raspberry Pi and GPIO (General Purpose Input Output) to power your speaker. But rather than using complicated terminology we treat the process of setting up your Raspberry Pi like a story with 15 to 20 words per step in the process. This Level 1 task is also documented in the companion book, which is filled with further details and history so that a teacher can elaborate to the class at their own pace.” The books are translated into Spanish, French, German, Mandarin, Hindi, and Arabic and are freely available from the Kano website.

The Kano project has produced some wonderful products that are sure to electrify the imagination of many children around the world, and its intention to share work with the larger Raspberry Pi community is to be applauded. Putting the Raspberry Pi in the hands of children and teachers is the true goal of the Raspberry Pi project, and the intuitive approach taken by Kano is sure to provide a useful and entertaining method of delivery. Expect to hear a lot more about this little project over the next few years. 📺

“Kano’s intention to share work with the larger Raspberry Pi community is to be applauded.”

CODE CLUB: THE NEXT GENERATION

Nine-year-olds are getting the chance to learn programming – and they love it. But for everyone to get a chance, more volunteers are needed, finds Richard Smedley.

If you're reading this magazine, then you most probably know the value of code, of people being able to code, and – perhaps most importantly – of understanding that coding is a creative activity, bringing frustration and joy, and the chance to fail safely. Now, imagine every child getting the chance to find that out at a young age.

Coding in schools became a hot topic a couple of years ago, famously with Google chairman Eric Schmidt giving the 2011 MacTaggart lecture at the Edinburgh Television Festival, where he said: "I was flabbergasted to learn that today computer science isn't even taught as standard in UK schools. Your IT curriculum focuses on teaching how to use software, but gives no insight into how it's made. That is just throwing away your great computing heritage."

This gave a valuable boost to campaigners in the tech industry who'd been saying the same thing for

more than a decade, but with a much more limited audience. Campaigners such as Clare Sutcliffe and Linda Sandvik, who came up with an idea for giving "every child between the ages of nine and 11 the opportunity to learn coding": Code Club.

The essence of Code Club is an after school club (in school, or another safe environment like a library), for

nine and ten year-olds, with a volunteer programmer using freely-licensed teaching materials.

Code Club is an organisation that's in the

right place at the right time, and its rapid growth has been in large part due to word-of-mouth advertising of how well kids take to its lessons. They're designed to draw children in, and get them being creative, learning the coding side by osmosis as they have fun.

To check this theory we asked a ten-year-old Code Club participant what she liked so much about it. She told us: "It's fun. You get to make games, like Flappy

"Imagine every child getting the chance to find out about the value of code at a young age."



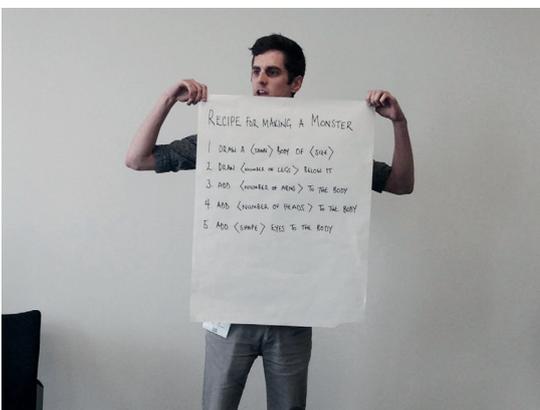
Soho Parish School and volunteer Simon Wharton, who started one of the first Code Clubs.

Bird. You get to learn lots of new stuff. There's lots of stuff to explore and it's fairly easy, because anyone can start it." She actually used the phrase "it's fun" several times, so we're happy to take Code Club at its word on this point.

Growing Up

Code Club's growth has been rapid – passing 2,000 schools by its second anniversary this spring – but it's matched by its ambition for the future. "One of Code Club's missions is to get Code Clubs into 25% of primary schools by the end of 2015, which is a huge target," says Sam Milsom, Code Club's General Manager. The demand is there, with 700 schools signed on that are still waiting for a volunteer. In response to this, Code Club has accelerated the release of all of its teaching materials under a Creative Commons licence: "which will allow teachers to use them themselves; it will allow parents to use them to teach their kids at home, and so on."

Terms 1 and 2 use Scratch to teach the basics of programming. Term 3 teaches the basics of web development using HTML, CSS, and a little JavaScript. Term 4 teaches Python. As we go to press, the possible content of term 5 and 6 is still under discussion, as Code Club is in the process of



Code Club Pro trains teachers for the new IT curriculum – but still keeps much of the fun that after-school Code Clubs have put into the subject.

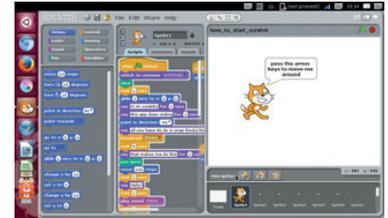
MIT Scratch

Scratch is a deceptively simple visual programming language and multimedia authoring tool developed at the MIT Media Lab. Platform neutral, and GNU GPL licenced, it enables learners to "mix" chunks of code before they've really learned what they're doing, giving instant results. Colourful graphic sprites and amusing sound effects add to the fun.

Yet this is no dumbed-down language: building on Squeak/Smalltalk roots, it's a fully-featured and powerful language, and as children go through projects, experimenting, they start to pick up code in a similar way to an earlier generation did by typing in program listings from 8-bit micro mags.

As well as all of the open source Code Club projects, the Scratch website hosts over 5,000,000 projects. Any young learner can click a button in the Scratch interface to upload their project and CC-license it – making Scratch perhaps the largest creator

both of new FOSS projects, and new FOSS coders. How many learners will go on to become Free Software programmers is not really the point (although some undoubtedly will); rather, Scratch immerses young people into a community where sharing (and re-mixing) code is seen as natural, and is a great antidote to more than a decade of schools teaching PowerPoint and Excel.



Right from the first lesson, kids are using event-driven, Object-Oriented programming.

appointing a full-time Curriculum Developer: "We are currently improving our web development curriculum and adding some exciting JavaScript stuff in. We are also writing lots more Python projects!" Laura Kirsop, Code Club's Managing Director told Linux Voice.

Materials should also get a rejig to allow less linear progression, so children can join more easily mid-way through term, and, Milsom tells us, "volunteers can pick and choose" what to teach. In fact, volunteers are already generating great material of their own, from those bringing in Arduinos and MakeyMakey boards (with the celebrated banana keyboard www.youtube.com/watch?v=rfQqh7iCcOU), to more "unplugged" exercises. One volunteer developed a coding role-playing game, Sam Milsom tells us, where "for one lesson the children didn't even sit at a computer. He developed an RPG and they'd speak out and act out the programming language. And he said it was incredible, because it got them to think about it in a different way, rather than just following instructions, and typing things in, they were really enacting it."

Milsom adds: "Our first two terms of Scratch were sort of crowd-sourced, and our third and fourth terms we had one person working on them. Obviously we use GitHub to host all of our projects, so the community do contribute, and we're hoping now that we've opened up our projects that will happen a lot more. Not just in terms of updates, but suggestions, and a lot of the community create their own projects, so again the opening of the projects will allow this to happen a lot more."

Volunteer effort

"We've got an abundance of schools wanting Code Clubs," Milsom tells us. The problem "is finding volunteers to go into the schools. We're hoping to get students from Manchester Metropolitan University



Back to school: now it's the adults' turn, as volunteers learn how to train the teachers in the new curriculum at Code Club Pro's first training workshop.

(MMU), and other tech companies – basically trying to get them to volunteer for us.”

We talked to Sam Milsom just before he spoke to a collection of Code Club volunteers, in Manchester: “We’re starting quite a big partnership in Manchester, between companies like CTI Digital, Manchester Digital Development Agency, MMU, and the Council themselves – and, of course, our current volunteer base in Manchester, which is the second largest hotspot of Code Clubs in the UK.”

This is a strong reflection of the tech industry and community in Manchester. London has the large companies, Silicon Roundabout, and Code Club's own offices, but outside of the UK capital, Manchester is the second largest tech city in Europe, with a vibrant start-up culture, and a strong Free Software and open hardware scene. But while Code Club has grown in the UK, groups have started to form in dozens of other countries, from the USA to Ukraine, and Code Club has responded with Code Club World.

Winning formula

Wherever they set up, Code Clubs follow the same simple formula of a volunteer programmer using the CC-licensed materials, working with a teacher (or other suitable supervisory adult in Clubs in libraries) who, though possibly lacking programming skills, has the experience of educating and controlling pupils that would be an intimidating barrier to many volunteers.

Code Clubs are free of charge for children and participating schools.

For many teachers, supervising the after-school clubs has been a welcome chance to see what all the fuss is about before changes in the UK national curriculum force them to learn something about coding. Milsom says that: “One of the benefits of Code Club – which is a happy accident – is that if you’re a volunteer and you go in, you have to have a teacher from the school present, and we realised that actually this is really a good thing: because it’s not just about the volunteers – industry professionals – teaching the kids. It also enables teachers to brush up on their skills, to get a good look [at teaching coding].”

Teaching teachers

The abolition of the current UK schools IT curriculum, which essentially taught children how to use MS Office, was announced during the early days of Code Club, whose close contact with schools allowed them to see teachers’ reactions to this, as well as to the announcement of the more rigorous, code-led curriculum being introduced this September.

“One of the reasons that we’ve opened up our projects is for teachers to have a look. I’m not saying our projects are going to be anything like the national curriculum – far from it – but I think, hopefully, it does demystify; they start to realise that coding in Scratch isn’t as scary as you think it is,” Milsom told us.

We suggested to Milsom that press coverage of the new curriculum as being only about “back-to-basics” coding had done it a disservice: “I’d completely agree with that. From day one we never expected that every child who went to Code Club would go off and become some great programmer who’d change the world. Not every child is going to grow into the next Sir Tim Berners-Lee. What’s interesting is that it goes beyond computational thinking: it’s simple problem-solving; it’s trial-and-error; and personally, I think there’s something about failure that is really important. Children aren’t taught that failure is okay, that failure is the key to success. They’re taught that you have to pass exams; you have to pass. And it’s my

Next, the World!

“We were becoming overwhelmed with emails from people in other countries, saying: ‘Can I set this up in my own country?’” Sam Milsom told us. Code Club World has now become a full-time concern for Code Club founder, Clare Sutcliffe, who told Linux Voice that CCW would: “create a framework to allow other countries to support local volunteers.”

Currently, frameworks of support for volunteers exist in some countries, such as Brazil and Ukraine. In others, such as India, there are several individual Code Clubs, but no national support mechanism. The model is the same everywhere: volunteer programmers, CC-licensed projects, and a

safe environment. Some things do change by country – in Canada there is no model of after-school clubs; pupils go home and eat dinner, then return to school for Code Club. Language is a barrier to growth outside the Anglosphere, but already Code Club’s teaching materials have been translated into eight different languages, and there are six more well on the way on GitHub.

Code Club World defines an active country as one with a team supporting 30 or more volunteers, and there are currently six at this level: Australia, Brazil, Hong Kong, New Zealand, Norway and Ukraine. The ambitious target, Sutcliffe tells us, is to get 100 countries with active communities by the end of 2018.



World domination? This time it’s a benign empire of educational volunteers spreading across the globe.

Volunteering

Before becoming Code Club's General Manager, Sam Milsom was its Volunteer Support Officer. He tells us: "We've got such an amazing volunteer community, we use Google Group forums, and have meet-ups, and some of the projects that the volunteers come up with off their own bat are incredible."

Go to CodeClub.org.uk and click on "Start a Club" and then the "Volunteers start here" button, and you'll find all the information you need, as well as hearing about the experiences of other volunteers. To work with children in the United Kingdom, you need an Enhanced DBS (formerly known as a CRB) check, and you'll need insurance to go in to schools. Code Club makes it easy to get these for free by linking up with STEMNET's STEM Ambassadors Programme, and encouraging you to sign up through your local STEMNET office. STEM Ambassadors is a volunteering programme for STEM (science, technology, engineering and maths) professionals and enthusiasts. After registering, you get a two-hour induction session on working effectively and safely with young people. After a few weeks your DBS check should be complete, and while you're waiting, as well as taking the time to read through Code Club's teaching materials, you can look on the Code Club website for a local school wishing to start a Code Club.

experience of seeing Code Clubs in action, that it teaches them that it's okay [to fail]."

The latest offshoot of Code Club is Code Club Pro, sending trained volunteers to teach groups of teachers how to deliver the new curriculum. As we near September's arrival of the new IT curriculum into schools, demand for CPD (Continual Professional Development training) in schools is likely to be immense. We asked Code Club Pro's Sophie Deen about this: "We're just scratching the surface of understanding that demand," Deen told us. "We're present in schools and hear back directly from teachers [who don't] understand coding or knowing how to get there.

Deen wants to show teachers that it's not just about code, but that computational thinking "can be used to enrich the way children think and learn in a cross-curricular way, covering core skills." In the first two months of Code Club Pro's existence, with no publicity, 850 schools signed up. "We know from teachers that when it [officially] launches there'll be a huge demand."



Code Club's quirky robotic graphics are an external reflection of a friendly, easy-to-access service.



If coding is the new Latin, then no wonder London Mayor Boris Johnson is getting behind it.

Many of the core concepts are already being taught, adds Deen: "Algorithms – instructions – are taught in English. Sequencing, basic logic and reasoning are taught already." Code Club Pro's materials are about: "Really trying to demystify the language and context of the new curriculum." Teachers are non-specialists, but good at teaching, and coding is no more difficult than music, which non-specialist primary school teachers regularly tackle.

Code Club Pro's materials aim to "demystify language and core concepts," showing that the computer language gives expression to ideas. "We want to help inspire the ideas," and to get teachers enthusiastic about the changes to curriculum. The materials should be online by the time you read this; if you'd like to volunteer to train the teachers, the process is a little more involved than volunteering to run a Code Club, as outlined in the box above. There's an online test, a Skype interview, then a day's training. "We have amazing trainers," says Deen: "Clever, good communicators and well-motivated."

"Code Club's teaching materials have been translated into eight languages, and there are six more on the way on GitHub."

Your turn

Have you always wanted to do your bit with passing on your enthusiasm for coding to the next generation, but been perhaps a little scared of facing a classroom of demanding and unruly children? Code Club's structure, with a teacher present at all times to work with the children, takes away this fear and gives you, dear reader, the perfect chance to spread some programming joy. The great materials mean that even if you are more of a system administrator than a coder, you can still teach – just read one lesson ahead, and you'll be fine.

You are wanted. You are needed. Step up to the task and you'll be appreciated, too. 📺

STEAMOS

AND THE

DEMOCRATISATION

OF GAMING

Discover how Valve singlehandedly changed the future history of games on Linux.

Valve is a video games company with some serious history. It created genre-defining games with its Half-Life series. It built a cutting-edge games engine used by its own titles, and then went on to dominate digital games distribution with Steam, long before Apple thought about walled gardens and app stores.

But none of this history has been played out on Linux: until relatively recently, Valve was entirely wedded to Microsoft Windows, the master system of PC gaming. This was due to several factors; Windows had been able to maintain a position it held since the end of the DOS gaming era, thanks to its DirectX games API. Almost

two decades of intense competition between the accelerated graphics hardware manufacturers had dropped prices and boosted performance,

and the Windows drivers for them worked well. Windows was actually a pretty good gaming platform.

But all that has changed with Windows 8, at least for Valve. Gabe Newell, Valve's co-founder and managing director, has said that Windows 8 is a "catastrophe", which is somewhat ironic considering that he worked for Microsoft for 13 years before Valve, and was, according to one interview, "the producer on the first three releases of Windows." But users are also reconsidering their commitment to Microsoft's operating system in the face of both user-interface and community challenges. Microsoft wants the Xbox

One to become its proxy for home entertainment, and its changing attitudes to core technologies such as DirectX and .Net have left many developers looking to broaden their asset pools and experience.

And then there's the emergence of independent development studios, many of which are now accustomed to developing for consoles, tablets, smartphones and PCs using cross-platforms tools and APIs. For them, Windows is just another platform.

Next-gen gaming

The next generation of gaming consoles have also helped shift priorities. The PlayStation 3, for example,

was notorious for its use of proprietary technology, forcing developers to learn techniques specific to a single (complex) platform, rather than the

cross-platform approach taken by most studios. Now that both the Xbox One and the PlayStation 4 are based on what are essentially standard PC components, the only differentiators are their operating system and development environments (and performance, of course). On the one hand you have the PlayStation 4 running an operating system based on FreeBSD and using all kinds of open source tools, and on the other you've got Microsoft with its latest incarnation of DirectX. The result is that it makes sense for most games development, for most games studios, to be cross-platform, enabling studios

"Valve's future, and potentially the future of PC gaming itself, is now dependent on Linux."

Valve and open source

Valve isn't a natural Linux company. It rarely responds to emails, and its approach to the Linux community is somewhat detached compared with that of some other companies or distributions. However, it has given the complete library of all Valve-produced games to both Ubuntu and Debian developers in recognition of the work they've done to build the basis of SteamOS, and Valve's own team are beginning to have an effect on the open source tools that they themselves are modifying.

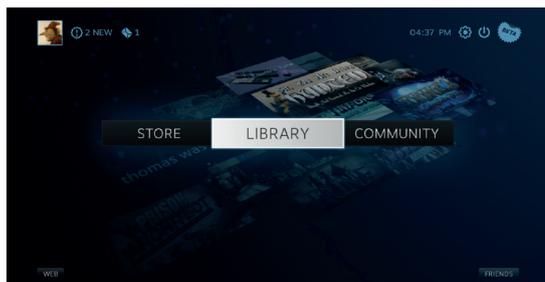
Its biggest contribution is the DirectX To OpenGL translation layer, which it uploaded to GitHub under the MIT licence (<https://github.com/ValveSoftware>). This neat bit of software helps developers migrate from Windows-only games, and could even help games for Microsoft's Xbox 360 console be ported to SteamOS/Linux.

Valve has also sponsored work on the Mesa project, the open source libraries that implement OpenGL with hardware acceleration, to help improve shader compilation time. The improvement should help the launch time of games using those shaders, and patches have already shaved 20 seconds off the launch time of Dota 2. Valve has also been helping make improvements to the XPad kernel driver. This is the driver used by many Xbox and Xbox-alike controllers, which have become close to being a standard on Linux and whose button mappings are the default in Steam. Valve's own Steam Controller uses the same driver, and the patches so-far submitted help with dynamic wireless device creation and LED feedback. Finally, despite their not being open source, Valve has been working with Nvidia to improve its proprietary drivers.

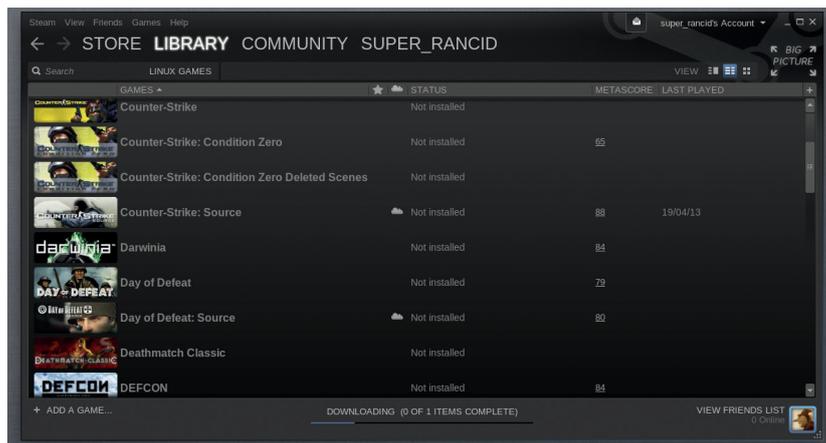
to release games for many platforms without the corresponding ramp in resources.

Valve has watched this shift in development priorities, and also watched Microsoft become more single-minded and controlling. It must have also seen the launch of Apple's app store and games portals, both on iOS and OS X, and noted that it will never be able to compete on a level playing field while in someone else's walled garden. The same could also be true, to a lesser extent, of Microsoft, and it seems likely that for both operating system vendors, the ideal situation would be one where their users could only install software (including games) through their own authenticated systems. And when your own app ecosystem is dependent on someone else's operating system, what can you do?

The perfect storm of a new generation of games consoles and the dawn of more restrictive and less



SteamOS isn't about improved performance, as there's very little difference. It's about controlling the ecosystem and competing with Sony, Microsoft and Apple.



competitive third-party publishing on Windows and OS X and forced Valve to come up with a dramatic change in direction. Until then, Linux hadn't been part of its history. But Valve's future, and potentially the future of PC gaming itself, is now dependent on what was an outsider and an underdog: Linux.

Steam client

The first whiff of Valve's changing direction came in early 2012. Some users reported that their Windows and OS X Steam clients included references to a non-existent Linux port, along with a few vague configuration files for the game Left 4 Dead 2. Most people cast these Linux appearances aside, as the rumours used to surface every few years regardless of fact. And until that point, the popular consensus had been that without obvious desktop growth, there was little advantage in a Linux port and only potential pain for Valve when it attempted to troubleshoot 200 different Linux distributions.

But those original rumours were confirmed on 16 July 2012, when Valve wrote a blog post called 'Steam'd Penguins'. It was used to straighten out the rumours, announce an 11-strong Linux Team (initially formed in 2011), and to finally confirm that they were working on both a Steam client and a port of Valve's Source games engine. Left 4 Dead 2 would be the first game. The blog also explained that Gabe had been interested in creating Linux versions for some time and that after conversations in the hallway sometime in 2011 those conversations led to the creation of a new team.

Valve has a rather unusual employee hierarchy because there isn't one. It describes this arrangement as a 'flat organisation' where employees don't report to anyone and are free to choose to work on whatever projects they think are interesting. In answer to the question "Why do I need to pick my own projects?" in the infamously leaked 2012 'Handbook for New Employees', the answer simply states "We've heard that other companies have people allocate a percentage of their time to self-directed projects. At Valve, that percentage is 100." Even Gabe Newell is himself described as follows in the handbook's glossary, "Of all the people at this company who aren't

The latest update to the Steam client includes game streaming for everyone and less OpenGL lag.



Steam and Linux are spreading the excellent Sir You Are Being Hunted across the world. Good show!

your boss, Gabe is the MOST not your boss, if you get what we're saying."

We'd imagine there was a constant background noise of Linux chatter at Valve. They already used it for their servers, and Gabe was probably vocal about where he wanted the company to go. At some point, the chatter resulted in a critical mass of opinion, and employees started pushing their desks together to create a new product, which became the Linux version of the Steam client. This *ad-hoc* formation eventually led to more resources and to the the 1,700 responses that appeared in the comments thread to the original blog post.

The Linux community were understandably very receptive to the idea that their operating system would finally be receiving a little premiere league gaming love. Many of us thought that the prospect of Steam finally coming to Linux was the last key in the puzzle and one of the last stumbling blocks for any operating system that wanted to consider itself as a

Steam Linux usage is still hovering around 1%, but there are many major titles in the pipeline and we've yet to see cheap Steam Boxes to compete with the PS4 and Xbox One.

MOST POPULAR	PERCENTAGE CHANGE	
	0.00%	0.00%
Ubuntu 13.10 64 bit	18.87%	-8.58%
Ubuntu 14.04 LTS 64 bit	16.17%	+16.17%
Ubuntu 12.04.4 LTS 64 bit	9.01%	-2.11%
Linux Mint 16 Petra 64 bit	6.75%	-0.85%
Linux 3.10 64 bit	6.59%	-0.56%
Ubuntu 13.10	3.84%	-1.60%
Other	38.78%	+1.53%
3 GB	19.76%	+0.96%
GenuineIntel	76.56%	+0.23%
2.3 Ghz to 2.69 Ghz	23.92%	+0.75%
3.3 Ghz to 3.69 Ghz	5.00%	+0.30%
2 cpus	49.30%	+0.19%
Intel HD Graphics 4000	6.60%	+0.45%

mainstream alternative to the proprietary alternatives of OS X and Windows. That blog post also outlined that initial support was going to be for Ubuntu only; which we all understood, because you don't want to deflect valuable resources from making a port to working out whether broken audio on Arch is down to ALSA, PulseAudio or OSS.

All hail the new gaming overlords

By December 2012, there was an open beta version of the Linux client, and while there were very few games – around 30 initially and growing to 50 by the time of the first official release in February 2013, Valve was having an effect on independent games developers. Many had started to consider creating a Linux port, as long as their cross-platform tools were able. But over the following summer, there was something more significant brewing. In the shadow of new console announcements from Sony and Microsoft, it seemed likely to most of us that Valve would attempt to enter the console market, and that perhaps its recent Linux manoeuvring would form part of a bigger plan.

The news eventually came in two slices at the end of September. The first was the announcement of SteamOS – a Debian-based cut-down Linux distribution that boots directly into a new Steam interface, and the second was an open hardware specification that it called the Steam Machine, along with a revolutionary controller. Unlike games consoles, there are going to be several different tiers for Steam Machines, and different manufacturers – some of which have already built and sold gaming PCs under the brand name. This means people can choose a specification according to their budget and gaming requirements, and even upgrade their own hardware.

Valve followed the hardware announcements with a lottery to send optimised prototypes of their own specification to 300 lucky Steam users. This was undoubtedly used by Valve to test their burgeoning Linux operating system, its Steam front-end and its hardware, drivers and update mechanism, a process that's still very much on-going. The launch of SteamOS and the promise of a new hardware platform was all that was needed for many major companies to start including Linux in their tier 1 gaming plans. Both Unreal Engine 4 and Unity 4 delivered Linux support, something which looked unlikely before Steam on Linux, and they're both making cross platform development much more viable for many developers. And with the prospects of Valve's Half Life Episode 3, a massively anticipated gaming title, coming to Linux on day-one, so too are other major studios considering Linux. There are now more than 431 Linux titles on Steam with more announcements than we can keep up with. More than the hardware and the software, this is what Valve has brought to Linux - a very real opportunity to create the best possible PC-based open gaming platform, which to many of us who love gaming, seems like a dream come true.

Installing SteamOS

It might be a moving target, but if you've got the luxury of a dedicated games machine, it's worth installing SteamOS onto a spare partition.

We'd recommend grabbing the ISO version of SteamOS. This is a 4GB file that can be downloaded from <http://repo.steampowered.com/download/SteamOSDVD.iso>. You then have the choice of either burning the ISO to a DVD, if your Steam PC is capable of booting from one, or transferring the contents of the ISO to a suitably large USB stick. The latter doesn't require any ISO conversion, such as with the UNetbootin tool, and the data can be written directly to the USB stick. The simplest method is `sudo dd bs=1M if=/path/to/dvd.iso of=/dev/sdX`, but you need to make sure that `/dev/sdX` is definitely your USB stick, as if you get this wrong, all your data will be overwritten. We'd suggest checking your system logs after inserting the device to make sure you get the correct device name. Before rebooting, we'd also recommend creating the SteamOS partition on your drives with a tool like GParted, as it's easier than doing the same thing through the SteamOS installer.

After rebooting, press the hotkey to open your system boot menu (usually F12). If you want SteamOS to be bootable from UEFI, make sure you boot into UEFI for your chosen boot method. If you want old-school BIOS booting, don't select a UEFI boot mode. Either way, the same SteamOS boot menu will then appear. Choose Expert Install from this menu, unless you want to give SteamOS complete control over your system. Then select a language, a location and a keymap. A few moments later, you'll see the 'Partition Disks' window. On our system, SteamOS had arbitrarily selected and suggested repartitioning our first drive, which is bad. Click on 'Go Back' and select the Partitioner again to remove any default choices. You'll then be able to double-click on your preferred

partitions. Change 'Use As' to 'Ext4', the 'Mount point' to '/' and enable the boot flag for your root partition. Click on 'Done Setting Up The Partition' to go back to the list and make sure only those partitions you're going to use are marked with a 'K' and an 'f' or 'F'. This is important, because the installer could change a drive you want to keep. When you're happy, click on 'Finish Partitioning And Write Changes To Disk'.

The base system will now be installed according to your wishes, and you'll get the chance to install the Debian desktop environment and the standard system utilities before the installer finishes, which we'd recommend for greater post-install flexibility.

After one more question your system will be rebooted, and with a bit of luck, you'll soon be presented with the SteamOS Grub bootloader menu, from which you'll be able to select SteamOS.

SteamOS should launch automatically, and on your first boot you'll first need to choose a language, agree to a EULA, change the screen configuration and your time zone. You can now use your Steam account details to log in and as with any other Steam client, you'll have to enter an activation code sent to your email address first. After the activation, you'll find yourself in SteamOS proper, complete with New Age background music and motes of OpenGL particles. While waiting for the official Steam controller, we'd recommend using Microsoft's Xbox 360 controller or Logitech's F310, as these work without any need to reconfigure the buttons. 

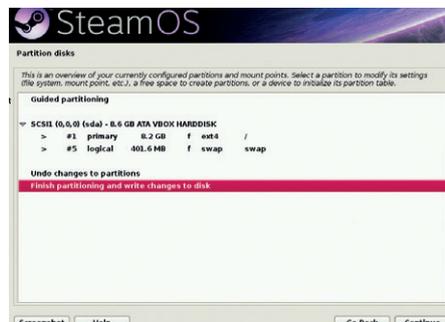
“You'll get the chance to install the Debian desktop environment and standard system utilities.”

Install SteamOS from ISO



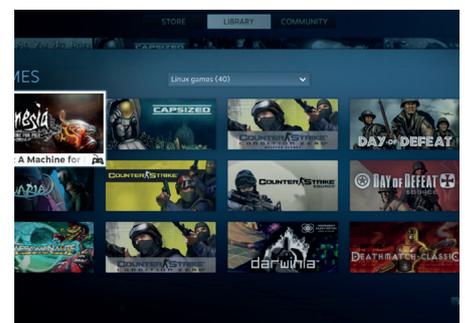
1 Boot it up

Burn the ISO to a DVD or copy it across to a 8GB USB stick. Boot from either UEFI or BIOS mode - SteamOS will now work with both. Choose Expert mode from the boot menu.



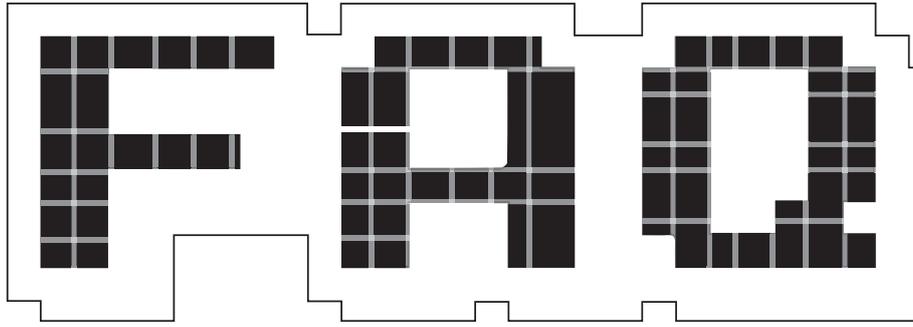
2 Partitioning

SteamOS recognises if a drive is empty and makes best use of it. Otherwise, be careful with its default configuration and make sure it doesn't overwrite your data.



3 Sign in

When it's up and running, validate your account and start downloading your games library. All Steam games are available to all clients, plus you can stream and share with family members.



LLVM/CLANG

Watch out GCC – there’s a new compiler suite in town, and it wants your crown.

MIKE SAUNDERS

Q Uh-oh – compilers. There’ll be a lot of acronyms and initialisms in this article, then...

A Actually, no. The technology behind LLVM and Clang is tremendously complex, but here we’re more interested in its practical applications – how it’s going to make life easier for developers, and make our applications faster and more reliable.

Q So what’s going on with this strangely named software?

A Right now, GCC (the GNU Compiler Collection) is responsible for compiling virtually all of the code in a typical GNU/Linux distribution. That is, it takes the human-readable source code for programs and turns it into executable files that the CPU can understand. GCC is arguably the biggest success story of the GNU project – it’s used everywhere, from supercomputers to embedded devices, and it can generate code for a huge range of CPU types.

“Some developers claim that the politics of the GPL scare away potential contributors.”

Q Great! So GCC rules – shall we just go to the pub now?

A Hang on a second. GCC is highly regarded, but it has flaws. The codebase is very complicated, and some developers claim it’s so messy that very few people can add new features to it, hindering progress. In addition, it’s hard to split up individual parts of the compiler, making it hard to integrate with modern IDEs (integrated development environments). Finally, the licence is an issue for some developers, who’d rather avoid the GPL and use something more BSD-like.

Q So this LLVM/Clang thing solves all these problems?

A Well, it’s a start. LLVM has been around since 2000, and Clang since 2007. Developing a new compiler is a gargantuan task, especially given all the languages, language features and CPU architectures in use today.

While GCC is still the *de facto* standard compiler in the FOSS world, LLVM/Clang is nipping at its heels, thanks to support from notable companies such as Apple. One testament to the maturity of LLVM/Clang is that it’s the default compiler in FreeBSD 10, replacing GCC. FreeBSD is well regarded as a conservative and highly reliable open source Unix flavour, so its adoption of LLVM/Clang was a major event in the compiler’s history.

Q You keep talking about LLVM/Clang – what’s with the funny name? Is it like GNU/Linux?

A No; it refers to two technologies that are parts of the compiler. In the olden days, compilers would simply read source code and generate equivalent CPU instructions on the fly, perhaps performing a bit of optimisation along the way. As time went on, it made more sense to split the part of the compiler that parses the source code away from the part that generates the CPU instructions. This makes a lot of sense, as it becomes easier to support more programming languages and CPU types. For instance, someone could add Fortran processing capability to the compiler without having to know about ARM and x86 CPU instructions. Or someone heavily versed in those CPU instructions can make optimisations without having to think about the high-level languages.

In order to separate these parts of the compiler and add a level of abstraction between them, you need an intermediate language. And that’s exactly what makes LLVM/Clang work. Essentially, Clang is a front-end for C-like languages (C, C++, Objective C), so it parses C code, includes header files, handles macros and so forth, and then generates some intermediate code. This code looks a bit like a mixture between assembly language

and a high-level language:

```
%result = mul i32 %X, 8
```

Here, the 32-bit integer variable `%X` is multiplied by 8, and the result is stored in the `%result` variable. This is not specific to any kind of CPU type, but is sufficiently low-level for LLVM to process. (That's where the name comes from – Low Level Virtual Machine.) So Clang knows C, and LLVM knows this intermediate language. LLVM then performs optimisations and tricks with the intermediate code, before generating CPU instructions for the chip of your choice.

Q **Wow! That sounds rather clever... But are other programming languages supported?**

A Yes – quite a few. You can chuck out Clang and replace it with another language front-end that generates the intermediate language. LLVM doesn't care. Right now there are front-ends for Ada, D, Fortran and other languages, and many of these front-ends were taken from the GCC codebase. And because LLVM is released under a BSD-like licence (so you can do what you want with it, providing you give credit to the original developers), it's making its way into various other compiler suites, IDEs and projects: www.llvm.org/ProjectsWithLLVM.

Q **Hrm, I'm not sure I like the BSD-style licence, whereby anyone can make closed source programs with LLVM inside. Wouldn't the GPL be better?**

A We're big GPL fans at Linux Voice HQ, so we know what you're saying. And Richard Stallman isn't especially happy about LLVM either:

"The existence of LLVM is a terrible setback for our community precisely because it is not copylefted and can be used as the basis for nonfree compilers – so that all contribution to LLVM directly helps proprietary software as much as it helps us." (Full message at <http://gcc.gnu.org/ml/gcc/2014-01/msg00247.html>)

Still, there are other sides to the argument. Some developers claim that the GPL scares away potential contributors due to all the politics involved, whereas a BSD-licensed compiler makes it simpler for

companies to add patches. It's a massive debate with a million points to be made, and GPL vs BSD arguments will rage for decades to come.

Q **Hokey dokey. Anyway, earlier you talked about LLVM/Clang making our applications faster and more reliable. How does that work?**

A One of the goals of Clang is to produce more informative error messages than those spat out by GCC, making it easier for developers to spot and fix bugs. Ideally, this should result in better code with fewer bugs.

<http://clang.llvm.org/diagnostics.html>

shows what the Clang team is doing in this direction. It should be noted that GCC is making progress in this area too, with colourised output in GCC 4.9 as an example. Competition is good!

Right now, LLVM and GCC are pretty much neck-and-neck when it comes to the speed of their produced code. GCC has a tiny edge in some benchmarks, but given that LLVM is a much younger project, it's impressive that it has reached the same level so quickly.

Because LLVM's codebase is simpler and easier to navigate than GCC's, the hope is that it'll be easier for new developers to add optimisations, and the more people that can get involved with the compiler's internals, the better. So while LLVM doesn't magically make applications faster now, if its development continues more rapidly than GCC's, it might outperform the GNU compiler substantially over time.



Every good FOSS project needs a mascot, and LLVM's robot is rather darn cool, we must say.

Q **Can LLVM/Clang compile everything that GCC can?**

A Not quite. GCC provides various extensions for C and C++, and because it has been the *de facto* standard compiler for free Unix systems for decades, many developers use these extensions. Clang's support is constantly expanding, though. In addition, there are efforts underway to make the Linux kernel compilable by LLVM/Clang, (see <http://llvm.linuxfoundation.org>). Right now it's possible to compile the kernel, but some external patches are required.

Q **OK, so how do I get it installed and test my code?**

A Few distributions include LLVM/Clang by default, although almost all of the big-name distros have the compiler in their package repositories – look for `llvm` and `clang`. Interestingly, some Debian developers are trying to build a version of the distro entirely with LLVM/Clang; of the 40,000+ packages that Debian includes, only 11.6% of them can't be compiled right now. As LLVM/Clang improves and becomes more compatible with GCC, this number will go down, and in a few years we might see mainstream distros compiled entirely with the newer compiler.

GCC is still an awesome compiler, and LLVM/Clang's presence has rejuvenated its development. With both teams scrambling to make faster, more informative and more reliable compilers, the future looks very rosy indeed. 🐉

PIRATE MONKEY ROBOT NINJA: PIMORONI

Fancy launching your own maker revolution? You could do worse than follow the example of one of its most successful offspring.

Perhaps William Blake was right. There may yet be a New Jerusalem built in the shadow of his Dark Satanic Mills.

In the north of England, the cities of Sheffield, Manchester and Liverpool are hosts to a new kind of industrial revolution, and it's one where any one of us can create anything. It's a revolution for makers and tinkerers and it's built on the success of the Arduino micro-controller, the Raspberry Pi, and of course, Linux.

Co-founded by Paul Beech and Jonathan Williamson, Sheffield-

based Pimoroni epitomises this new community and maker attitude. Paul is the designer who won the competition to create the (now famous) Raspberry Pi logo, and he made the original laser cuttings for the 100,000-selling Pibow case at his local hackerspace. Jonathan is a software engineer and startup veteran. He's unfazed by PCB design, wave soldering machines and arrays of laser cutters.

We met them to discuss their beginnings, their remarkable growth and what's next for the company in the heart of the Maker Belt.



“The ancient Greeks had absolutely no concept of graphic design. When they were making their alphabet they just didn't think about branding at all.”

LV Did you guys know each other before you got together to create Pimoroni? Had you discussed doing something like this before?

Paul Beech: We'd done some startup stuff before, but we've kind of levelled up to one another haven't we?

Jonathan Williamson: Yeah, we'd known each other for about eight years probably.

PB: I think we got to know each other when we were doing startup things and doing our first freelance things, starting our own first business together, all that kind of thing. We've crossed paths many times and got on together when working and hanging out.

LV Did the Pibow and the design for the Raspberry Pi logo come about at around the same time?

PB: No, the Raspberry Pi logo was created a few months before the Pi

came out. It was part of their...

LV Their competition!

PB: Exactly. I saw the BBC article and I was following the project after that because I thought, 'Yeah, this is gonna to change everything'. And then they had the logo competition and it was one of those things where I was so inspired, I said 'This is going to be so huge, it needs a kick-ass logo'. It took me about an afternoon.

LV An afternoon!

PB: Yeah, well, it was an inspiration, because I believed in it, so the berry popped out of something I thought was good. The most important thing was dropping the Pi concept. The ancient Greeks had absolutely no concept of graphic design. When they were making their alphabet, they just didn't think about branding at all.

LV But they knew about Bucky balls [the shape of the C60 Buckminsterfullerene molecule]...

PB: Absolutely. Aristotle probably worked it out and then thought 'Nah, no practical application'. So, that was around August or September the year before it came out.

LV There were a load of delays and it came out about nine months after we all expected it to.

PB: Yep, there was all the getting it to manufacture, which they blogged brilliantly, and people were frustrated about it, but actually getting that level of detail from the process was really great as well. That's another thing that made us think 'Yeah, great project'. They would tell you exactly where things went wrong and why, which is very cool. But, anyway, it got released and people started getting it in their hands. There



was conjecture when the cases were coming out, they were functional, pretty boring. And there was one case that I think Adafruit made, which slotted together, but I got one and bits of it broke off and it wasn't quite there. I mean, there were good reasons for it but it just like 'Oh, this isn't right'. So then I said, 'OK, I'm gonna make a case'.

The Adafruit case was nice because it wasn't injection molded, so it's something you can make yourself on a laser cutter. And Adafruit is very open with its designs. They'll let you download that kind of thing and let you do it yourself if you want. The main

“Getting that level of detail from the Raspberry Pi launch process was really great.”

problem is that the perspex has terrible tolerances, so if you're relying on those kind of jointed connections, they're either going to be too tight or too loose and you'll very rarely get something bang on.

JW: And the perspex is quite brittle in single sheets.

PB: So for laser cutting, I went to Fab Lab in Manchester [this place looks brilliant – www.fablabmanchester.org] and kind of bounced off there with trying to get in and use the laser cutter. It's a very good place and they have various kinds of open policies on what you can come and do, and how you use the laser cutters. And if you use them on Fridays and Saturdays, they'll offer you feedback on your design. They make it open for people to learn. It was trying to get onto the laser cutter and get a bit of help getting started that was hard. At the same time, Access Space

in Sheffield have got a laser cutter, and I knew them a little bit, so I kind of walked through the door and went 'Hey, can I use your laser cutter?'

LV **Do you get people coming around to you now saying 'Hey, can I use your laser cutter?'**

JW: Yeah, a fair bit! And we try to let them if we can.

LV **We probably shouldn't put that in the interview!**

PB: It's something we'd like to do more of but it's trying to make it so that it's easy for them and easy for us.

JW: Like scheduling it and having a window where we say just come in and use the laser cutter. Generally it's fine, but sometimes we're in a massive crunch and we just can't deal with anything else at that moment.

PB: We do want to have a bit more of a

hackspace vibe, but we'll just see how it pans out with the new place. So I ordered a stack of swatches of perspex and it came in a portfolio of all the different colours, and I didn't think anything of it. Then John was around the flat one day, saw it, and he knew all the colours from the top to the bottom and said 'Ah, that one's quite nice'.

JW: That was my entire contribution.

PB: Then it was about 10 hours over a couple of weeks, Access Space

“Open Source allows us to look at all sorts of stuff and learn from it, which is great.”

[<http://access-space.org>] being very helpful, and we had a prototype!

JW: The first prototypes were hilarious though, because they would be ones all in black or whatever because it was just a single sheet, but there would only be half a lid because the sheet had run out. Because from an A4 sheet, you would get about eight or nine slices on it and half of the lid.

PB: You had to use your imagination.

JW: Just to fit around the Pi, yes, the dimensions are pretty good. It's 99mm by 66mm by up to 33mm, so it's complete *2001: A Space Odyssey* 3-2-1, except it's a bit thinner so it looks kind of nice. Probably a designer's trick.

PB: One of the other things were the bolts, because originally they were metal and the nylon just felt a lot more playful.

Were you always going to use bolts to hold the case together?

PB: I don't think we were going to do anything to be honest. We were just buying small things and putting them together and saying 'Yeah, that looks better than it did last time'. It made sense with the layers. Because they're not at right angles to each other, you're not going to dovetail them or join them, and that was the problem with the other cases. It wasn't the first thing to be layered, but it was tricky getting them to go around the ports, because the ports on the Pi are pretty crazy.

JW: So solving that, you've got to have splits on the edges where the port connectors come out. And, if you just cut rings and holes, you'd just have a stack of useless pieces basically. So you've got to structure it internally to make sure it bolts together.

PB: This is where I think we worked together really well. There's one phrase I like which is if that two people always agree, one of them is useless. If two people always disagree, both of them are useless. And me and John are kind of a perfect middle ground. So I'd bring the case to John and say 'Look, look, you see this John, I think it's alright, there's this, this and this', and he'd say 'And that, that and that doesn't work, and that's a bit crap, and it's leaning like this' and I was kind of 'Yeah, I was kind of ignoring that', but it was good that he pointed it out.

So you were still serious with one another at that early stage.

That's good! I think that's very telling.

JW: Yeah, we were both quite serious and nerdy about stuff.

PB: We used to read everything on the internet twice a day, all of it, and we'd know what we like, we know what works and what doesn't, and try to match it as much as possible.

JW: We both have quite well-formed opinions on what we like and don't like.

PB: Which sometimes means we don't agree on something, so we just don't do it because neither of us can get it to that point where we both like it, even though one of us feels it would improve the end result.

JW: It's difficult, especially working with materials where a small change can make a massive difference to the end result. And you don't know it until you try it, which is why having the equipment around is nice for prototyping stuff.

So where does the circuit design come from then?

JW: Well, I used to do a bit of electronics when I was 15, but then I didn't touch it again. That was mostly from playing with the Adafruit stuff really, wasn't it. You were into it (Paul), so I got into it.

PB: We both did electronics at school and when we were young, but that's a reasonably long time ago for us.

JW: We discovered electronics through Adafruit Industries, SparkFun and companies like that.

PB: The thing is, electronics is much more fun than it used to be. Because it used to be just analogue electronics, which is all about knowing the equations and doing the maths. You could do interesting stuff, but it took a lot of groundwork.

Whereas these days, you can grab amazing sensors, hook them up and, although there's proper ways of doing things too, but you can get things to work without very much effort. And we learnt a lot more about the crap way of doing things, which is kind of where it gets a bit boring again, but actually you can start getting results through very quickly and easily.

You can spend about \$30 and have an accelerometer and a colour sensor and a temperature sensor, and direct it



Laser cutters cutting plywood smell like burnt toast. Significantly nicer than PCBs.



Jonathan: "When I was a kid I used to get told off for staying on the computer until 2AM writing Assembler on our 386."

to an Arduino or a Pi and start playing around with that in software, and it's amazing. It makes it a lot more like software than traditional analogue electronics. So we got into it from the easy angle, and then we've spent the last nine months to a year really learning it properly.

"We know what to do when things set on fire. We've got fire extinguishers..."

Open Source allows us to look at all sorts of stuff and learn from it, which is great. We released the Pibow under Creative Commons non-commercial licence for the same reason. We don't want people cloning us just so they can write a book, because the people who do that aren't people who contribute back to the community. But we do want people to, if they want to use a laser cutter they've got access to, make their own crazy extensions, make up their own colours, or do whatever they want. Or if they're using it in school, we want them to be able to do that.

JW: Children at school cutting their own cases, that's what it was all about really; that was the whole point.

PB: But someone can do better than us probably. So we learned from open source stuff, seeing how it was done. John has spent probably the last eight months to a year just basically hacking away, making mistakes and then making it better, and then improving it and working out the other really esoteric things that made it better.

LV **But the scale that you're working at now, seems a world away from messing with a stack of coloured perspex. How do you go from Arduino and temperature sensors to selling 100,000 units?**

PB: You've just got to learn quickly.

JW: We're lucky the Pibow gave us the money to buy equipment.

LV **But how do you go from Arduino and temperature sensors to the scale your operating at at the moment?**

JW: Laser cutters are just a really good introduction. They're a relatively safe industrial machine that's not very expensive either. So learning on lasers was great. Once we'd done that, then things like the wave solder seemed a lot less scary. You know, it's just a series of robots, or it's just a big heater and a load of molten metal. That's not too bad now. We know what to do when things set on fire. We've got fire extinguishers. That's all fine. So taking the first step meant that the next step seemed a lot less scary.

We forget how daunting it was. That first time, two years ago, when I walked into Fab Lab and I'd just seem people doing smoke and stuff, I thought that was pretty scary stuff, I need someone to hold my hand a bit. And Access Space was the place that did that, and got over these first few humps. After the first hour, my complete mindset had changed from 'Whoa, scary fire' to 'Ooo, fire!' I can make anything now, as long as it's kind of flat.

LV **And made out of slices, which I guess is anything pretty much.**

JW: Yeah. Paul had Autodesk 123D, and they have a tool now and you can do 3D stuff in there, and it will cut in slices for you. And it will cut in so you're kind of tessellating slices. So if you've got a model of a dinosaur, it'll cut it so you can laser cut corrugated cardboard

that you can slot together into the model automatically. You have to do a bit of finicky, but it's there.

LV **So, as you've spent the last eight months or so learning the machines and the electronics, does that mean we're about to get a great new product?**

JW: There might be some things in the works! We're working on the final retail version of Picade at the moment. Basically, the main reason we're doing that is because we're doing a mini and a maxi double product.

PB: We're delivering the last 50 odd units now, waiting on the last batch of LCD monitors. They go out the door, then we've got some software stuff to do. Just support our users how we said we would. Obviously getting the hardware out there and then updating the software is kind of accepted now. So, yeah, we've just been heads-down getting the hardware fixed.

LV **You were the first Kickstarter project in the UK [with the Picade – a desktop Raspberry Pi arcade machine], so it must have been quite a new concept for most people – buy now, take delivery when it's ready. How have your backers been?**

PB: They've been really really exceptionally patient. Because we were looking at grumblings and that's when we failed to update them and keep them in the loop. So that was totally our fault.

JW: There were literally times when, and it's not an excuse, but we'd be sat in that workshop at the darkest times of that Kickstarter, both sat staring at the Kickstarter page, at the update form and think we have nothing to say. You know, I know what's going on, but none of it's of help to anybody.

PB: We want to please people. We don't want to say 'Yeah, this has been a delayed, we haven't quite sorted this out, we've got nothing useful to say on this and we've got no idea how to do it'. We didn't want to do those updates.

JW: The best thing is if we'd have done the updates, it would have been better.

PB: But everyone's been super supportive, and the few that have grumbled have been much happier when we've responded, and just told them what's happening. 



YOUR AD
HERE



Email andrew@linuxvoice.com to advertise here

LINUX VOICE REVIEWS



Andrew Gregory

Has been given a box running Ye Olde Vista from 2006. Elementary OS, we need you now.

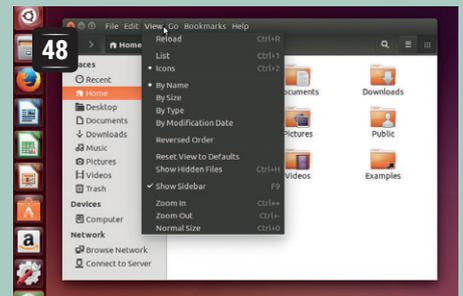
The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

On test this issue...



Acer C720 Chromebook

Google, thanks very much for this smart, light laptop. But please, can you stop following me now?



Ubuntu 14.04

'Pretty' used to be Ubuntu's killer feature. Now it's a smooth interface, a reliable core, heaps of software and a silly name.

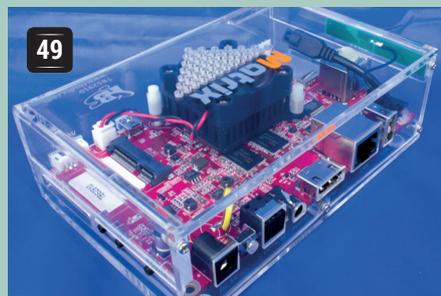
There's an uneasy meeting of cultures when Free Software and money intersect. At Linux Voice, we plan to give some of our profits back to deserving projects. Personally, I'd like it if one of those was the Electronic Frontier Foundation, as I think that's a body that really understands the issues presented by the changing tech landscape.

Then there are companies like Canonical. The work of organisations such as Debian is there for all to take and build on, and that's exactly what Ubuntu does. If it makes some money out of doing this, great.

Ubuntu/Debian is the best analogue to Kano/Raspberry Pi that I can come up with. To some, Kano is piggybacking on the work of a charitable foundation, packaging it nicely and selling itself as something that's going to revolutionise computing (forgetting the fact that the revolution is happening already, with or without tasteful orange websites).

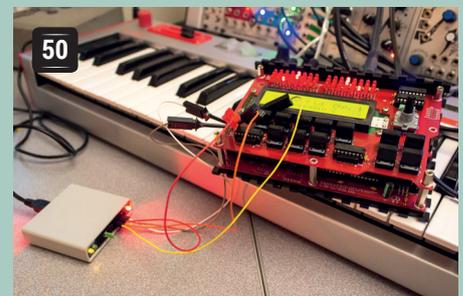
But it's more helpful to see Kano as a partner. Just as some grumbled when Ubuntu made Debian easy to install (how dare they!) and put a nice Gnome interface on it (the temerity!), the Kano project is packaging the Raspberry Pi. It's taking Linux out there into the wider world, and as such is helping to spread the message. Greed is good.

andrew@linuxvoice.com



TBS Matrix

On the plus side, this ARM box makes a great platform for a DIY smart TV. On the downside, you need Windows to use it...

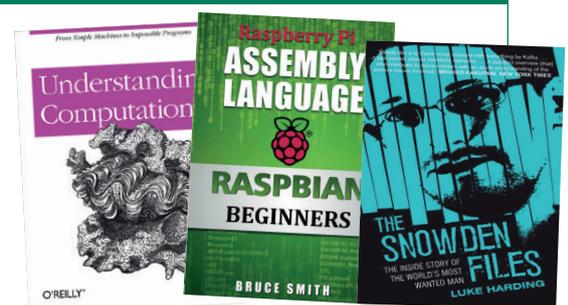


BitScope BS10

Generate waveforms, monitor the data coming out of your devices and intensify the hardware hacking fun.

BOOKS AND GROUP TEST

The deeper you get into Linux, the more time you'll find yourself at a terminal emulator. If you're just using your PC for Facebook and Twitter, you won't care, but for everyone else, a good terminal is essential, so we've tested a handful of the best and found something there for everyone. What caught our eye in this issue's book reviews was *The Snowden Files*. Snowden, by his illegal/patriotic/heroic/treasonous actions, has given us all cause to think about our security. For that, he deserves a slap on the back; or maybe something more vigorous...



Acer Chromebook C720

There's never enough Google in your life. Says Google. So it's a good job that you (and **Graham Morrison**) can install something else on this great little Chromebook.

DATA

Web
<http://www.acer.co.uk>
Developer
 Acer
Price
 £199

SUNSPIDER 1.0.2 BENCHMARKS

C720 with Chrome OS:
 411.4ms
Firefox/Ubuntu LTS:
 315ms
Intel 2.4GHz i5 (4258U):
 168.2ms

Not many of us would have thought that a laptop with little more than a web browser would be useful. How can you build things? Or edit anything other than text? Or put together a magazine, or play games?

However, portable computing has become more divergent than we could ever have imagined when we wanted our netbooks to do everything. We're now productive with smartphones and tablets, despite neither being capable of running a virtual machine nor a compiler, and the typical Chromebook is an extension of that idea. It's a low-powered laptop that boots quickly into Google's Chrome OS, a Linux-based operating system that's basically nothing more than the Chrome browser running in fullscreen mode. There's no support for applications other than the web-based ones you can download through Chrome. There's no command line, and there's very little scope for customisation.

What you do get is online and offline access to Google's carefully crafted suite of productivity applications; word processing, spreadsheets and presentation creation alongside YouTube, Maps, Keep and anything else you install through Google's web store. It all works extremely well, and the lack of the usual clutter that comes with an operating system helps to keep you focused – that is, until you install Command & Conquer. There's very little to be said about Chrome OS, which is a good thing for certain



There's a USB 2 port and a USB 3 port, plus an SDCARD reader – but there's only 16GB of usable local storage.

scenarios. There is a small task manager bar at the bottom of the screen, and a launch menu on the right for the Google apps and other apps you may install. Configuration is through Chrome's settings panel, which is augmented for Chrome OS with options to change the background image, network management, and touchpad control – both for sensitivity and for reverse (labelled 'Australian!') scroll settings, which is a mode we're ashamed to admit we now prefer.

Chronos

Acer's C720 is better than the average Chromebook. It has a 1.4GHz Intel Celeron CPU descended from the current Haswell range, which is a massive step up from the ARM CPUs used by many chromebooks. It's a processor that has far more in common with the CPUs found on more general-use laptops and desktops, even if you can't take advantage of its particulars from Chrome OS. It does, however, help Chrome OS feel extremely slick. Wireless internet resumes within seconds of lifting the lid, and it deals with complex websites, WebGL, HD Adobe Flash and HTML playback with ease.

There's very little local storage – around 9GB to play with, but Google obviously wants you to use its own cloud product and Chrome OS plugs directly into your Drive account, as well as offering a deal on 100GB cloud storage for 2 years. Google's video conferencing app, Hangouts, is a good example of whatever hardware acceleration Chrome OS must be leveraging, as we had our best experience on the C720 after trying many devices, from the Nexus 5 and



Acer's Chromebook is surprisingly light, yet despite that it feels like it could withstand the bumps and knocks of travelling to a thousand geek conferences.

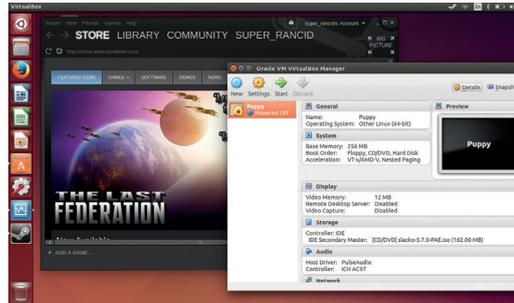
Running Ubuntu

What many Linux users really want to know is how the laptop performs if you install a different flavour of Linux into it. We installed Ubuntu 14.04 with relative ease, but this was mainly thanks to ChrUbuntu, a series of scripts that automatically handle the re-partitioning and downloading of the required packages. All you need to do is go through the slightly convoluted procedure of entering developer mode and making sure that every command you type looks sensible.

The main problem with running Ubuntu rather than Chrome OS is that a default installation requires around 4.5GB of space, leaving you with only a few precious GB for your own data if you keep Chrome OS installed alongside – which we'd recommend. Ubuntu ran brilliantly on the C720, and running a full-fat Linux distribution on the Chromebook felt very liberating. Overall performance, for a 1.4GHz CPU, was fantastic. We even installed Steam to play a few games, and the 3D acceleration in the CPU was more than adequate for many of the less demanding games in our collection. Unfortunately, it's the 2GB of RAM that's the biggest limitation. There's theoretically a 4GB version of the same laptop, but we've not been able to find UK stock, but that would make life a lot easier for other distributions.

Thanks to a VT-x enabled CPU, we even had VirtualBox up and running with a couple of Linux distributions. The biggest

limitation was storage and RAM, but it's still an impressive feat and genuinely useful for running small or server-oriented distributions. Battery performance under Ubuntu was about the same as Chrome OS, but there were a few hiccups that would probably flummox a lot of users. The touchpad stopped working with a kernel upgrade, for example, and we couldn't get Wi-Fi to resume from suspend or the USB 3 port to work without a little script tinkering.



Ubuntu runs brilliantly on the Chromebook – it was possible to run OpenGL Steam games and VirtualBox.

Nexus 10 to quad-core 16GB desktops and laptops. We also had almost two weeks standby time and a typical battery life of 7–8 hours, depending again on whether we launched Command & Conquer.

We were disappointed by the screen, as this hasn't moved with the times. It could be argued that a 1366x768 resolution on a 11.6" screen is acceptable when all you're doing is running a full-screen browser, but we're growing increasingly used to the high DPIs found in phones and tablets. It also lacks punch when compared with IPS displays and has a woeful range of viewing angles. The screen, along with the gun-grey plastic case design helps to make the C720 feel rather more utilitarian than it should. We also wish there were more options for storage. We understand that the tiny SSD is there to keep costs down and perhaps to limit its use outside of Chrome OS, but we'd like the option of adding more. Watching a film or managing anything more than a few albums is going to be impossible without a great data connection. Fortunately, while we've not tried it ourselves, it looks relatively straightforward to replace the SSD with a

larger unit, even if it does invoke the use of a small plastic wedge. We did like the keyboard, however, and the Chrome OS specific keys – backwards, forwards, task switching, and refresh – made us miss their inclusion on many post-Windows 98 keyboards.

You can work and type very effectively from the Acer and we were able to write many words for this very issue from its keyboard. It's also light enough (1.3kg)

and small enough (19mm thick) to throw into almost any bag, and as long as you can offer some kind of networking connectivity, it's the perfect travelling companion for people who need to do some work.

We like the C720 a lot, mainly because of its price, portability and relative power. At just less than £200 in the UK, it could be the perfect laptop to give to family or friends who aren't too confident with computers, where you know the limited options of Chrome OS are going to be all they'll ever need. But it also makes an extremely good low-cost travelling companion, especially if you put another Linux distro onto it. If you can find a compatible SSD, it makes a great machine for hacking about on and for throwing into a rucksack for LUG meetings or hacker sessions. ☑

“The C720 Chromebook makes an extremely good low-cost travelling companion.”



Many games are playable though the Chrome web store, and it's surprising just how powerful the average web browser has become.

LINUX VOICE VERDICT

Excellent value for money and perfect for non-technical relatives, corporate deployment and expert Linux tinkerers.



Ubuntu 14.04

Unity and kernel tweaks make up for what is otherwise a fairly low-key release, says **Mike Saunders**.

DATA

Web

www.ubuntu.com

Developer

Canonical Ltd

Price

Free to download

Canonical has its fingers in many pies at the moment. Go over to www.ubuntu.com, for instance, and the first thing you're greeted with is a big "the cloud platform of choice" message, accompanied by some blurb about OpenStack. Hover over the Download link and the first options are Cloud and Server – almost as if the desktop is an afterthought. On the other hand, Canonical is still pushing Unity, its mega-convergence interface that will (eventually) run on your desktop, phone and TV.

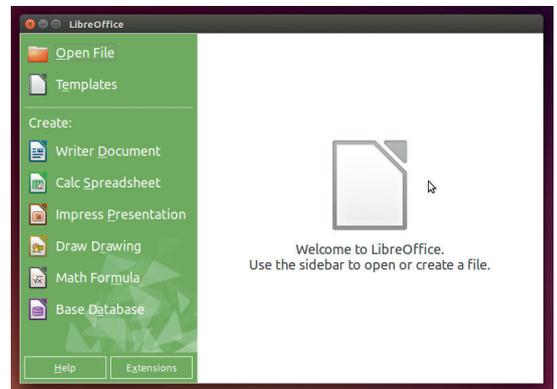
Ubuntu 14.04 is an LTS (Long Term Support) release, which means it will receive five years of updates and is designed for use in large enterprises where even the smallest changes can cause massive disruption. So don't expect world-shaking new features here: there's no Mir or Unity 7, and by and large the distro is a bunch of small (but useful) updates rather than anything revolutionary.

For starters, it's built on kernel 3.13 with support for TRIM. This boosts performance on machines with solid state drives, and given that these are becoming more common, it's a welcome improvement. If your machine has Nvidia Optimus graphics, you can now switch between the high-power Nvidia chip and Intel's battery friendly alternative – although you'll need to restart X for the changes to take effect.

GUI fiddlings

Interface-wise, much work has been done on Unity to make it more usable with HiDPI displays, such as those on the Retina MacBook Pro and Chromebook Pixel. Menus and launchers now scale better without looking teensy-tiny on super high-res displays. And

Global menu haters rejoice: it's now possible to have menus back in individual applications (albeit in their title bars).



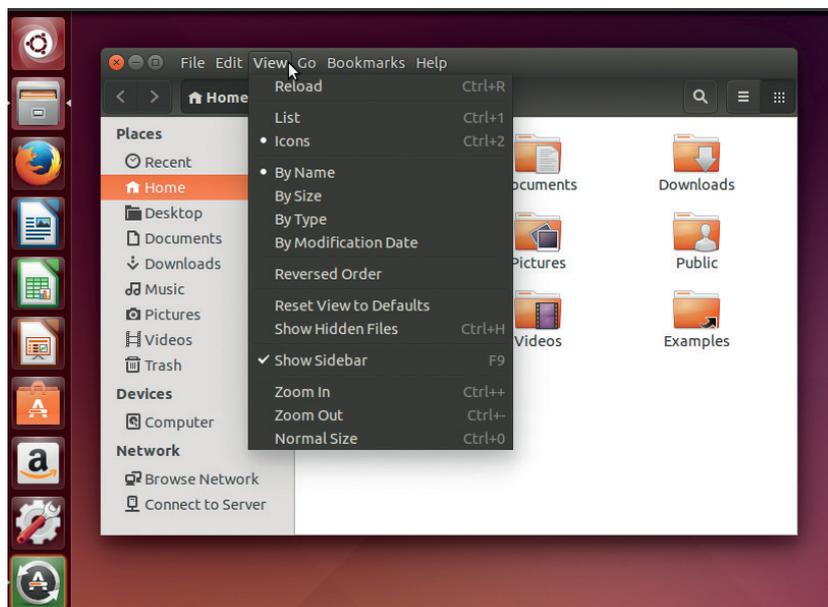
LibreOffice 4.2 sports a new start screen that's more attractive and useful than in previous releases.

speaking of menus: if you utterly hated the global menu bar, whereby all windows showed their menu entries in the top-left of the screen, Mac OS style, you can now disable this and revert to a traditional format via Appearance > Behaviour > Show The Menus.

It's still a bit unusual: menus are displayed in the program's titlebar, and you have to hover over the titlebar before you can see them. You can still click and drag anywhere on the titlebar (even when the menu is displayed) to move a window. When you resize a window in Ubuntu 14.04, its contents are updated automatically – there's no yellow box like in previous releases.

Unity has taken a lot of flack over the years, but Canonical has persisted with tweaks and fixes in every release, and it's now at the point where we're pretty happy using it. Software-wise, Ubuntu 14.04 ships with LibreOffice 4.2.3, Firefox 29 and Thunderbird 24. Rhythmbox remains the default music player, while Totem is used to play videos. If you're looking at this LTS release on a server, you'll find Apache 2.4.7, MySQL/MariaDB 5.5.37, PostgreSQL 9.3.4 and PHP 5.5.9. And for programmers there's GCC 4.8.2, Clang 3.4, Python 3.4 and Perl 5.18.2.

There's nothing that screams "install me" in Ubuntu 14.04, but it's worthy of an LTS release, and Canonical hasn't tried to rush ahead with anything that isn't ready. It's easy to ask for more in a new distro version, but when a company has to support it for five years, it makes sense to take a more conservative approach. **L**



LINUX VOICE VERDICT

Performance boosts, Unity tweaks and software updates make this a solid, if not hugely exciting, release.



TBS Matrix

A little Linux machine for building smart TVs – **Ben Everard** unglues himself from The Jeremy Kyle show long enough to take a look .

The Matrix is designed as a little ARM computer to run Tvheadend, the TV streaming server for Linux. It comes with a minimalist version of Linux, pre-loaded with the drivers for several of TBS's DVB receivers, as well as the software to use them (Tvheadend and XBMC). In theory, this means it should be really easy to create your own Linux-powered smart TV, and record all the shows you want.

The Matrix is well designed for this purpose. The quad-core Freescale i.MX6 processor with 2GB of RAM provides enough horsepower to decode the video streams in Tvheadend and display them in XBMC. There's 16GB of internal storage so there's space to record a limited amount of TV even without external storage, but if you want to expand it, there are plenty of options including a SATA port and a mini PCIe. To get the video out, there's an HDMI port, and for audio there are 3.5mm jack and optical ports.

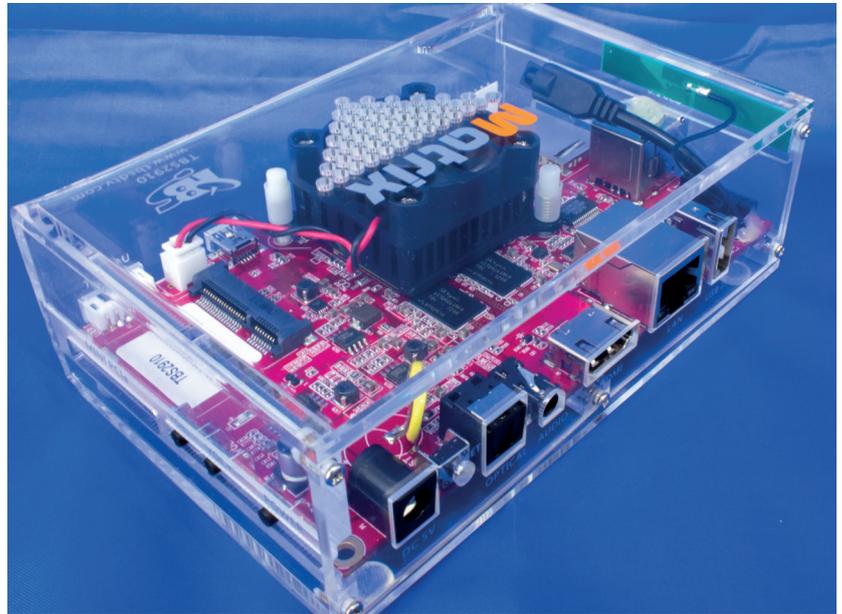
More than a box

The hardware looks good, feels solidly made and works well. However, this alone isn't enough to make a Linux-powered smart TV: there's also the software side of things to consider. Tvheadend is a challenging beast at the best of times (and using it with the Matrix and a TBS USB DVB tuner is the best of times). It takes a little prodding to get everything set up, and there's not much documentation from TBS to help you with that. The Tvheadend wiki is a good place to start (<https://tvheadend.org/projects/tvheadend/wiki>). As long as you're using a supported DVB receiver, all the necessary software will be installed. It shouldn't be beyond a reasonably technical person, but it can take a few hours, especially if you haven't done it before. Once it is set up, Tvheadend integrates really well with XBMC either running on the Matrix itself, or on a separate computer.

In addition to the Matrix's own minimalist distro, there versions of Android (4.2) and Ubuntu (11.10)



XBMC performance is fantastic and could make the Matrix one of the best frontends you could buy.



The clear plastic box feels solid to us, and provides good protection for the Matrix, while showing off the circuitry to glorious effect.

available to download. Unfortunately though, you'll need a Windows machine in order to push them onto the internal eMMC storage. The reliance on Windows will put some people off, and it's a bit of a shame that only Ubuntu 11.10 is available as we much prefer the 12.04 LTS version.

The Matrix is capable of running either of these OSes without any real sluggishness. Android comes with the full Play store, so you've got access to all the usual games. It would be nice to have the ability to dual boot, because we'd really like the option of switching between Android for games and MatrixTV (TBS's custom Linux distro) for TV, but this isn't possible at the moment.

The 16GB of storage means you can run it without any expansion should you wish, which is a big advantage over some ARM boards. We would also expect this storage to be a bit more durable as we've had some issue with SD cards on other ARM boards.

Over all, we'd say that the TBS Matrix is good hardware that's let down by the software. It's a good option for building a PVR, but don't expect it to be completely straightforward. 

DATA

Web
www.tbscards.co.uk
Developer
TBS Technologies
Price
£159.99

LINUX VOICE VERDICT

A handy little ARM PC that's most at home as streaming and recording broadcast video.



BitScope 10

It's an oscilloscope & analyser in a tiny case with Linux and Raspberry Pi support, and it's completely taken over **Graham Morrison's** life.

DATA

Web

bitscope.com

Developer

BitScope Designs

Price

£160 (approx.)

What's an oscilloscope? It's a way of measuring small variations in voltage. You've probably seen them in their CRT lab coat incarnations, plotting sine waves or the harmonics in the words from an alien overlord. They're almost essential for anything other than the most basic of electronic tinkering, because they enable you to monitor changes over time, unlike a voltmeter for example, which simply shows the voltage.

What's an Analyser? It's similar to a scope, but it's designed for capturing multiple digital signals at once so that you can see the relationship between them. They're useful for reverse-engineering the output from old chips or decoding digital protocols from the signals themselves.

If you're any kind of electronics tinkerer or hobbyist, connecting chips to an Arduino or playing with I2C on a Raspberry Pi, oscilloscopes and analysers are essential for troubleshooting. But more importantly, they're a lot of fun.

Lights, action...

BitScope's BS10 is both an oscilloscope and an analyser. It's relatively cheap, and that's because instead of the screen and controls of standalone units, you get a block of exposed copper pins held within a tough extruded aluminium case and software for the functionality. The BS10 is tiny – 67x64x17 mm – fitting neatly within its own handy carrying case that also contains the small clipped wires you use to connect the pins to the things you want to measure.

The 26 pins provide a huge range of facilities, with their assignments helpfully labelled on the underside of the unit. There are two inputs for the dual-channel digital oscilloscope, labelled A and B and marked by removable green and yellow plastic jumpers. To



It's a little fiddly to use if you've got large hands, but you can also attach standard probes or a differential add-on for a more professional solution.

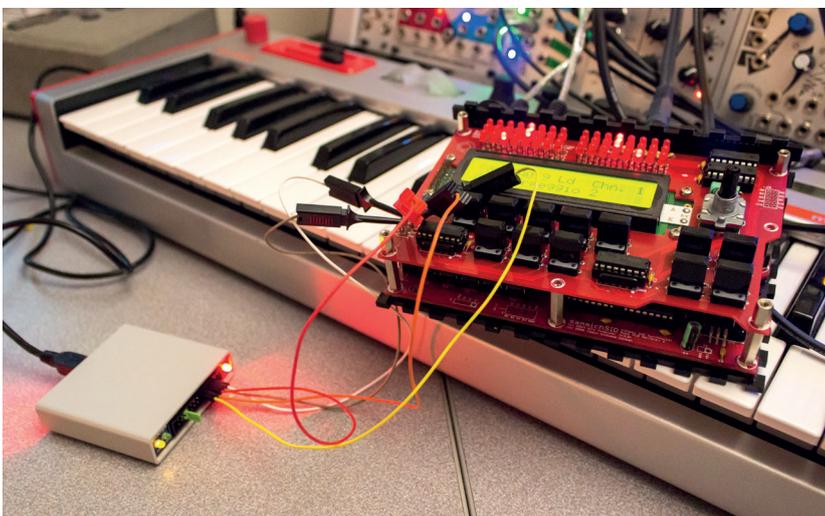
the right of these are digital input pins for the logic analyser, and to the right of these the waveform generator output. There are lots of separate ground pins for each input, plus 3V and 5V outputs and general-purpose pins.

A USB port on the rear powers the device when connected to your Linux box. No drivers are required, as the device is driven by the FTDI support built into almost any Linux kernel, but you'll need to install one of the applications from the comprehensive software suite to be able to start using the unit.

It's the job of the software to handle the functionality that you'd expect from the controls and screen of a more expensive unit, and the principal application is called 'DSO'. We installed the Deb package on Mint with a single click, but there are no real dependencies, so you should have no issues with a different distro. RPM and (32-bit binary) downloads are available, and we also tried the Raspberry Pi ARM package, which worked flawlessly. BitScope has blogged about the amount of effort it's put into optimising the Raspberry Pi version, and it's easy to see why. Connecting the BS10 to a Raspberry Pi makes for a convenient package that will work exceptionally well in an educational setting, especially if you're programming the Pi to send signals being monitored by the BitScope. You could even add a low-cost touchscreen to create a DIY hardware oscilloscope.

DSO is an overwhelming application, not helped by a lack of hardware-specific documentation. It's the

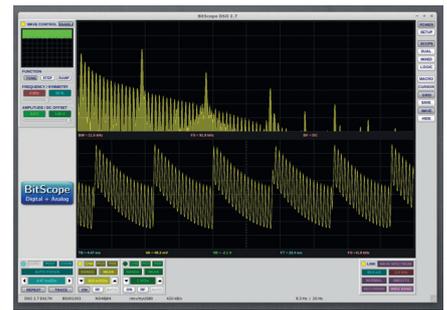
The Bitscope feels like a large toolbox of functionality. Here it's probing the inner poetry of the output from an 8-bit SID chip.



An oscilloscope for the Raspberry Pi

BitScope recently launched a new model, sometimes called the Micro and sometimes called the BS5. It's tiny a piece of hardware based on the BS10 we've reviewed here, only squeezed onto a long thin PCB that's covered in a resin to make it waterproof. On one end is the USB connector and on the other end is a 10-pin subset of what you find on the model 10, making this device cheaper (£120). The remainder of the specification seems identical, including the number of analogue and digital channels and all of the functionality that's unlocked through the software. It's a perfect partner for the Raspberry Pi and helps to explain why BitScope has spent so much time making sure its USB products work well with a device known to have a USB bottleneck.

One consequence of this, apart from the tinkering opportunities of programming your Raspberry Pi while you monitor the output, is that it makes a cheap network-attached monitoring station. One of the tools in the software suite is 'BitScope Server', which after being installed, is simply executed on the command line. The device needs to be plugged into the USB port of the server and the server tool running. Anyone else on the network can then run DSO to connect. It works exactly as it would were the hardware connected directly, and what surprised us most was that you could connect to the same device from multiple instances of DSO on different machines. This could be useful for lab or teaching scenarios, as well as remote monitoring.



The RPi drivers have been enhanced to make best use of its limited USB bandwidth.

software that's used for all of BitsScope's hardware, and it suffers a little from trying to do too much. It can display a single-scope channel, dual channels, the logic input and a mixed combination of all of them, alongside a waveform generator and a spectrum/phase plot of the analogue inputs.

The challenge is to enable and manage these features, and it all starts in the top-left. Without a trigger, the hardware doesn't generate data. A trigger is what captures and recognises a cycle or a frame from an input waveform, enabling you to monitor the input in real time (if you've got repeat enabled), or saved as a single cycle on the display. From there you can accurately measure the inputs, either manually with the cursor or automatically, and adjust the timeframes/voltages for capture and scaling. If you've used an oscilloscope before, this will all be familiar territory, as too will be the units and terms used throughout. But for a beginner, using DSO for the first time will leave you feeling a little like how you must feel having read through this paragraph.

Sine your name

We can only hope you'll persevere, because there's an incredible amount of power here. Far more than even on a cheap hardware oscilloscope. The 100MHz bandwidth of the BS10 is enough to for real-time repetitive capture, and you can adjust scales separately for both channels, as well as trigger inputs from the other channel. You can adjust code and parameters as you're monitoring the signal, and we were able to play with all kinds of cross-modulated audio signals and voltages in ways you just can't without dedicated hardware. We did have some difficulty mastering the triggers, but the waveform output – which includes sine, square and triangle waveforms – is a good way of testing your setup against differences in external hardware.

The other major mode in DSO is the logic analyser, used for displaying binary data captured on the digital inputs. We attached ours to a PCB connected to a SID chip and used DSO to monitor the various bits of control data going into the chips. This is a complex

area, but the eight input channels were more than adequate at showing what was happening across many pins at once, and with these inputs reportedly working at 40 million samples per second, they should be able to decode even complex protocols, especially on older or off-the-shelf components. You can cross-trigger the digital inputs from the analogue input (and vice versa) and use a trigger mask to look for a specific state or digital value, all of which helps make best use of the limited buffer space within the DC10.

From an educational perspective, there's a lot more to the analyser mode, as it enables you to see what your software is actually physically doing and how devices communicate. This isn't easy to accomplish with other methods, and can also lead into all kinds of engineering and design territory not easily covered by software alone. And while there are many, many other features we haven't mentioned, perhaps the API is the most significant, as it enables you to access many of the same functions within DSO only through your own Python/C/C++ code, which could have all kinds of uses for your own projects, tutorials and even installations.

Oscilloscopes and logic analysers have always been expensive and are usually considered luxury items by most hackers. But they make the impossible possible, and they're a lot of fun. The BS10 packs so many features into its small case, it's difficult to know where to begin. It has great Linux support, works well on a Raspberry Pi, and can take you from Arduino tinkerer to an electronics and engineering wizard. Just don't rely on the (non-existent) BS10 manual. 📖

“The BS10 packs so many features into its small case, it's difficult to know where to begin.”

LINUX VOICE VERDICT

Fantastic value for such a powerful and versatile device, suitable for so many different users.



Raspberry Pi Assembly Language – Raspbian Beginners

Mike Saunders don't need no silly high level languages, fool.

Given how much people love poking around with the Raspberry Pi hardware, it's surprising that there aren't more books on low-level programming for the dinky device. Indeed, books on assembly language for any platform are hard to come by. This 260 page tome is available as both a paperback and a Kindle edition, the latter of which can be read via a web browser at <http://read.amazon.com>.

Raspberry Pi Assembly Language... introduces ARM assembly language pretty well, although it dives straight into the details fairly quickly: you learn about binary arithmetic before you've even printed a line of text on the screen. We tend to prefer a more hands-on approach where you learn things by doing, and not just reading. Still, the chapters are well written and dense, and cover using libc, disassembling C programs, and working with the GPIO pins.

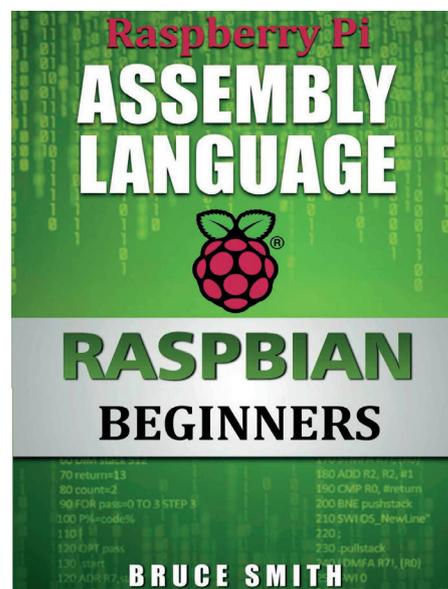
The code samples are well commented, and there are appendices for the ASCII character set and Linux kernel system calls. In all it's a solid read, tackling a notoriously difficult subject with confidence and making the reader feel like he/she is making good progress through each chapter. We'd just like to see more practical code samples, especially early on when tricky concepts are being introduced.

LINUX VOICE VERDICT

Author Bruce Smith
Publisher BSB
ISBN 978-1492135289
Price £14.99 (print), £5.97 (Kindle)

Great value, and thorny topics are explained well. Could do with more code snippets though.

★★★★★



Mike's head explodes with potential when there's more than 640k of base memory.

Understanding Computation

Graham Morrison found this book ideal for understanding his own code.

This book isn't ideal for morning reading before you've had a coffee or two. Nor is it perfect for a mid-afternoon browse, when your lunch is having maximum effect on your glucose levels. It's a book to be read while you're wide awake and engaged, because it doesn't hold back.

Within the first 20 pages, you're dealing with the semantics of expressions and trying to get your head around the Ruby code used to illustrate most of the ideas, and that's after you've read a page about the meaning of 'meaning'.

Automata for the people

Despite this being a book aimed at programmers "with little or no formal training in computer science," the last time we read a book like this was while studying computer science. This isn't a cookbook of programming recipes. It's a comprehensive overview of the theories behind computing

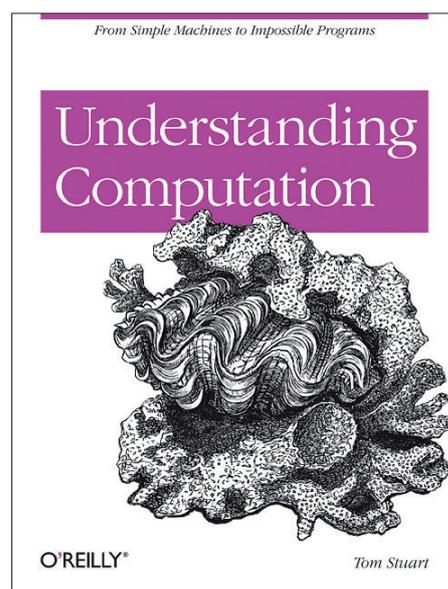
and computers, by example. It really does make you think about how you approach your own projects – what statements and expressions are really doing. There's a lot about finite states and Turing machines, at one point building compute units out of Conway's Game of Life. But this is all theory. There's nothing you'll immediately be able to take away and use, and if you want to apply what you've learnt to your own projects, you'll find their foundations broken, forcing yourself to start from scratch.

LINUX VOICE VERDICT

Author Tom Stuart
Publisher O'Reilly
ISBN 978-1-449-32927-3
Price £26.99

An intense and complex book about the theory many of us take completely for granted.

★★★★★



Almost five pages consist entirely of the characters `-, >, {, [x, y, l, m n, o, p,], and }`. If that sounds like fun, you'll love this book.

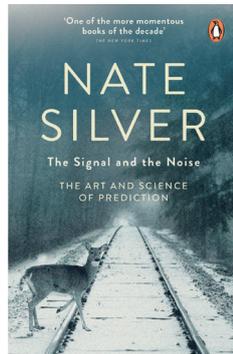
The signal and the noise

Ben Everard never trusted the TV weatherman. Now he knows why.

Nate Silver earned his reputation as a forecaster by correctly predicting the results of the 2012 US presidential election for every state in the US. In this book, he looks at how people make predictions and forecasts. It's a guide to help regular people understand the perils and pitfalls of trusting the forecasters.

The reader learns why some TV weather forecasts are less accurate than the raw forecast data the TV stations base their forecasts on, and how to tell if an economic forecast is accurate.

In an age where everyone seems to be trying to collect big data, and analyses it for traits about us, the world and everything else, understanding the difference between signal and noise, and how they affect our ability to correctly understand the world (and therefore predict what will happen), is now more important than ever. This book is important because it coolly looks at a number of areas where data is used



Get it, read it, then laugh at the certainty of so-called experts in the news.

to predict the future, and investigates whether those forecasts have been successful, and why.

LINUX VOICE VERDICT

Author Nate Silver
Publisher Penguin
ISBN 978-0-141-97565-8
Price £8.99

Essential reading for anyone who has ever wondered about why we trust experts.



The Snowden Files

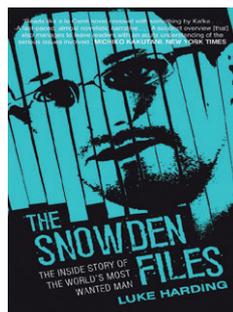
Ben Everard wonders if James Garner will star in the TV series.

Few people actually know that much about him. When he appeared in the news most frequently, he had risen to fame so quickly that reporters struggled to find details of who he was or what he did. At the same time, information about his whereabouts and travels was often kept guarded.

Luke Harding obviously has access to some of the people involved in the reporting of the case, and *The Snowden Files* details exactly what happened as Edward Snowden went from high-school dropout to NSA contractor to the biggest whistleblower in history to man-on-the-run.

It is, perhaps, a bit mis-named because the main focus of the book is not the files he acquired from the NSA, but the man himself.

The book paints a picture of Snowden that's broadly in line with how Snowden has been portrayed in most unbiased media, so don't expect to find any shocking details, but the story is well



Edward Snowden still faces a life of uncertainty.

written and engaging. In many places it reads like a spy story, which in a slightly unusual way, it is.

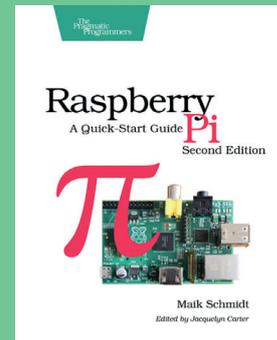
LINUX VOICE VERDICT

Author Luke Harding
Publisher Faber & Faber
ISBN 978-1-78335-035-3
Price £12.99

We found the book mesmerising, and we're sure most people interested in the events will do too.



ALSO RELEASED...



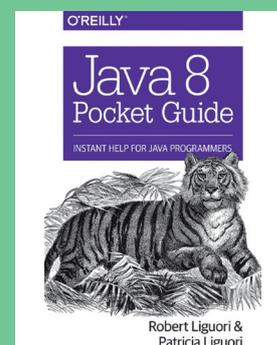
Other books about using the Raspberry Pi are also available.

Raspberry Pi – A Quick Start Guide
This is the second edition, and as things have changed so much, and the Pi has become so successful, we'd imagine there are plenty of new things to write about. This is still a book aimed at the complete beginner, however, so we'll let you know whether it succeeds.



The only things we've ever disliked about this series are the big heads on the cover.

Head First JavaScript Programming
There have been a few misses, but mostly, O'Reilly's Head First books are brilliant. They make technical subjects fun and immediate in a visual web 2.0 way that may be getting a little tiresome, but not quite yet. JavaScript is the perfect subject matter for this style too.



Java – who'd have thought it would still be kicking around ?

Java 8 Pocket Guide
If our Android coding tutorial has got you wondering about Java again, then this small reference book contains quick access to details like naming conventions, types, statement and blocks. It's perfect if you've used Java in the past, or use another OOP language and need a quick reference.

GROUP TEST

Marco Fioretti opened more shells than you want to know to help you discover what terminal emulator is right for you.

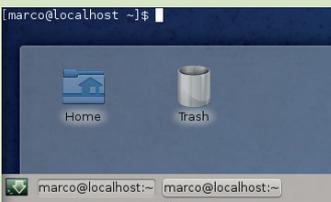
On Test

Eterm



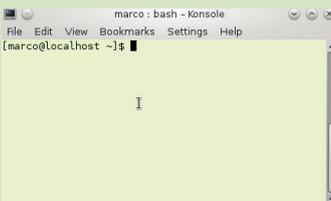
URL www.eterm.org
Version 0.9.6
Licence BSD
The original emulator of the Enlightenment window manager: is it still up to the task?

Guake



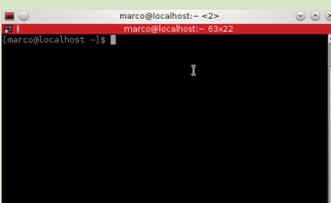
URL www.guake.org
Version 0.4.4
Licence GPLv2+
An "on-demand" emulator that hides in your system dock and only opens when called.

Konsole



URL <http://konsole.kde.org>
Version 4.12.4
Licence GPLv2 and GFDL
The official terminal emulator for the KDE desktop environment.

Terminator



URL <http://gnometerminator.blogspot.com/p/introduction.html>
Version 0.97
Licence GPLv2
Ascetic power.

Terminology



URL www.enlightenment.org
Version 0.4.0
Licence BSD
Made for Enlightenment, it conceals a hidden trove of features.

Terminal emulators

Relics of a long-gone era, or effective tools?

Terminal emulators are rectangles on a screen that let users communicate with the local computer, or a remote one, much like in a generic chat session: you type some commands as text, and the computer answers by displaying other text, or launching another program. All GNU/Linux distributions include, or let you install very easily, a range of terminal emulators.

Such interfaces can, however, look boring or intimidating, especially for people who always used touchscreen or mouse-based computing environments. Do they still make sense in 2014?

The answer of this Group Test is a resounding yes. It does make sense to know terminal emulators, and the available choices in this field, for a very simple reason: mouse or touchscreen interfaces are gratifying and easy to learn

because they are like baby gestures: whatever you want, be it a document on your drive or a movie from the internet, you point your finger at it, just like you did to get toys when you were a toddler.

The obvious problem is that gesture-based interaction is the most effective way to go in many cases (think drawing software) but not in all ones. The reason is that you can only "work" with what was already visible on screen. Learning to talk is a daunting, apparently pointless task for babies, but there is a reason why all those who can, eventually do it: it's the only way to explain exactly what you want, or to provide a complex description of a problem. Terminal emulators provide access to the same communication system used among grownup humans: speech. That's why we're delighted to present you with five great choices.

"Graphical interfaces are easy to learn because they are like baby gestures."

THE CRUCIAL CRITERIA

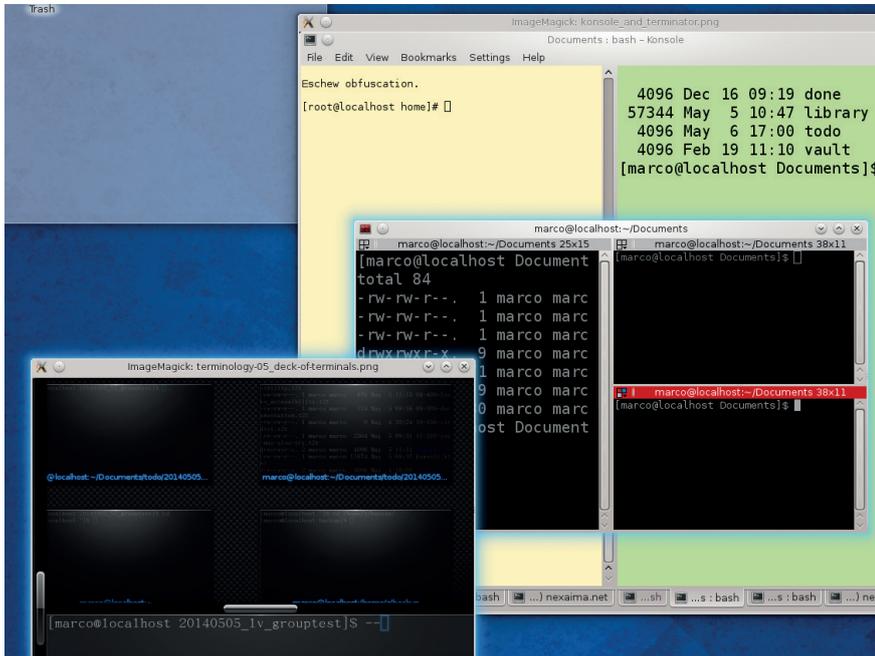
We wanted to show you how many different choices of terminal emulators you have under Linux, including some you may have never discovered otherwise. We also wanted to give you choices as easy as possible to install and try.

Therefore, we looked for emulators that are rich in features and easy to install, normally available as binary packages, but that are not the default ones in the most popular Linux distros.

This, and the fact that we only have space for five products, are the only reasons why we left a good emulator like Gnome-terminal out. It's included in the basic versions of (at least) Debian, Ubuntu, Fedora and all their cousins. Besides, it has all the main features discussed in these pages, and is somewhat hidden in one of them. In other words, you really have no excuses to not try it at least once.

Multitasking

Is it easy to manage many terminals simultaneously?



Konsole, Terminator and Terminology can shuffle terminals in more ways than you'll ever need.

By now, you know that terminal emulators enable interaction with computers more or less like written speech. As it happens in other realms of life, it is often necessary to carry on, or at least keep open, several conversations in parallel. Four of our emulators make it very easy to do so. The exception is Eterm, at least in the version tested on Fedora 20. Guake has tabs, like Konsole, but with fewer settings.

The tabs in Konsole work very much like those of Firefox and other web browsers. You can right-click on the name of each tab to close it, rename it or move it to a separate window. We also like the possibility to automatically assign a different colour to each new tab.

A dedicated panel in the Konsole configuration interface (Settings > Configure > TabBar) lets you hide the Tab Bar, put it on top or bottom of the window and place in it dedicated buttons to add or close tabs. Our preferred function of that panel, however, is the one that sets where all new tabs should appear, that is, at the end of the bar, or next to the currently active tab.

Terminology can split its own window and each sub-window both vertically and horizontally. In addition to that, you can

right-click in any sub-window and select "New", to open a new terminal exactly over the already existing one. Moving from one terminal of the same sub-window to another is easy to do but hard to spot. When you create more terminals in one sub-window, the emulator activates a very small terminal switcher in its upper-right corner: the default colour scheme of Terminator makes it hard to see, but you can just click there to move from terminal to terminal.

Terminator is so flexible from this point of view that if you find yourself using all its functions together... it probably means you're working too much. First, there are both tabs and multiple levels of vertical or horizontal window splitting. Second, there is another feature that is even cooler, even if many people will find no use for it. The coloured rectangle in the left-hand corner of the Terminator status bar opens a menu in which you can define groups of terminals, so that everything you type in one of them is broadcast to all the others.

VERDICT

Eterm	★★★★★
Guake	★★★★★
Konsole	★★★★★
Terminator	★★★★★
Terminology	★★★★★

Installation and configuration

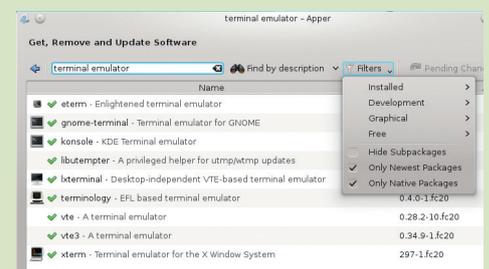
Is it complicated to get these emulators up and running?

Due to the criteria with which they were chosen, none of the emulators considered for this Group Test should give you any trouble. Unless you use some little-known distribution, you should easily find binary packages with the software manager of your GNU/Linux system. In any case, here are a pair of issues that are worth mentioning.

The first is that, while emulators are small programs, they may consume much more disk space than you expected, especially if you install them as packages. Konsole, for example, will bring with itself most of KDE, even if you had no plan to ever use any other part of that desktop. Guake and Terminator may do the same with parts of Gnome.

Eterm and Terminology are both products of the Enlightenment window manager community. They have the same "problem" as the other three – their dependency on Enlightenment libraries and other components. However, they will generally consume less memory, and much less space, than their competitors.

A final word of advice: depending on your distro, you may find that some minor feature of an emulator doesn't work as documented (sound effects, background configuration and similar) if you run it under a different window manager or desktop environment than that for which it was primarily designed.



The software managers of all the main Linux distros include all the terminal tested here.

VERDICT

Eterm	★★★★★
Guake	★★★★★
Konsole	★★★★★
Terminator	★★★★★
Terminology	★★★★★

Customisation

What makes a good terminal?

For us, a good terminal must have at least these characteristics: configurable keybindings for all the main operations, ways to define custom commands, and support for automation. The latter feature consists of being able to memorise and load, automatically or on user demand, complex combinations of many terminals, each with its own settings.

Konsole and Terminator do practically everything we just mentioned, through assorted plugins and support for user-defined profiles. A profile is a set of configuration parameters for a single terminal, which is given a name (eg "root profile" or "web server profile") so that it can be loaded automatically, when the emulator is launched, or on demand.

The point of profiles is to make your terminal emulator do as much housekeeping as possible for you, by running predefined commands every time you load them. Let's assume that you always need to have three terminals open at all times: one for checking email, one for working remotely on your web server and one to execute all other commands and generic scripts at the prompt. You can define an email profile that, all by itself, before you even see the window, starts the Mutt email client; a web one, which automatically connects via SSH to your server; and a generic profile for everything else. Each profile may also load a completely different configuration.

Konsole and Terminator can also split their window in any combination of sub-windows. Terminator makes it quite easy to save these "layouts", with different profiles for each terminal. All its keybindings can be reconfigured, or individually disabled.

The other emulators have fewer features, but still enough that it is impossible to mention them all. Eterm has search and run boxes, plus many scrolling options. Terminology provides configurable "helpers" – applications to open all kinds of files, showing them inline if possible. Besides tabs, in Guake you can easily set the default interpreter for your terminals to bash, nologin and several versions of Python.

VERDICT

Eterm	★★★★★
Guake	★★★★★
Konsole	★★★★★
Terminator	★★★★★
Terminology	★★★★★

General behaviour

Can you live with them day-to-day?

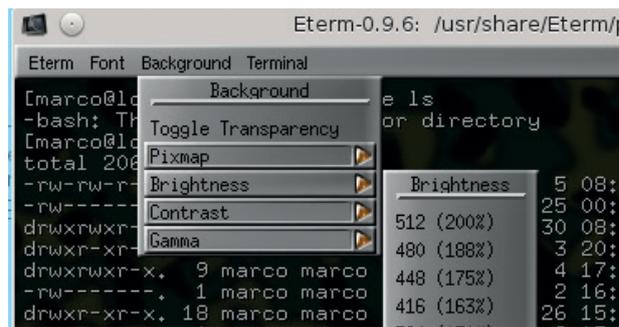
A large part of what we think of as the user interface of a terminal emulator is actually the result of its interaction with other, more or less independent programs; specifically the individual commands that we run inside it. No matter how you tweak the emulator itself, you may have to adjust the locale

settings of your distribution, as well as the configuration file of your shell, to make everything look and work just as you like: prompts, history, escape sequences, audio or visual alarm bells are just the main examples of what we mean. Keep this in mind, when thinking to how to use each of these programs.

Eterm ★★★★★

Eterm was conceived as the "Enlightened terminal emulator for the X Window System". It seems well suited both for expert users who want a lean and mean terminal, and for beginners looking for something fast, but with an unusual (shall we say "vintage"?) look and feel. Working in Eterm feels like running an Xterm (the main Unix terminal emulator of the 80s)

redesigned for 2014, and we mean this as a compliment. You can customise every detail of how Eterm looks and works, and tune its memory consumption with command line switches. If you have the patience, that is: we hope we're wrong, but the default looks and strain on the eyes of Eterm doesn't seem to encourage newbies, which is a shame.



Eterm has a pleasant vintage look, and enough visual parameters to set that it takes time to find the best combination.

Guake ★★★★★

The other emulators of this Group Test all aim to make long terminal sessions as efficient and comfortable as possible. Guake, instead, is "a drop-down terminal for Gnome", designed to be invisible, except when you need a prompt quickly, but just for a few minutes. That's probably why you can set the height of the Guake window, but not its width. Once started,

Guake sits in your system dock and appears (or disappears) in a flash whenever you press the F12 key. By default, due to this design choice, Guake is configured to "Stay on Top", but you can disable this behaviour in the General tab of its Properties window. A KDE version of the same emulator is available at <http://yakuake.kde.org>.

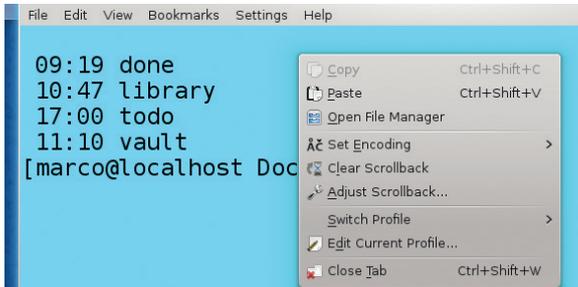


Setting an image background in all the tabs of Guake is a snap. Whether it's worth it is another question...

Konsole ★★★★★

It's an obvious thing to say, but we're going to say it anyway: Konsole works and feels just like the rest of KDE. If you have the time and patience to try all its options, and configure them to your taste, you will find yourself with a nice, very powerful and still unobtrusive tool.

However, even if Konsole were the only KDE component you'll ever launch in your computer, it would still work and present itself in the most discreet way possible. Clean menus, bookmarks, readable fonts out of the box... Konsole has many features you will like.

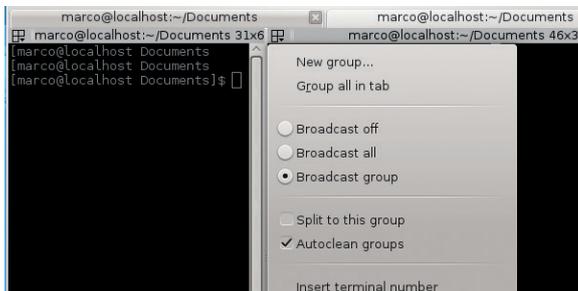


The default looks, fonts and other setting of Konsole may seem dull, but they are well thought-out and don't get in the way.

Terminator ★★★★★

The Terminator home page greets visitors with the half pompous, half cryptic slogan "The robot future of terminals". After just a few hours of usage, you see why for yourself. Technically speaking, this program is a tool to host multiple Gnome terminals in one window, as efficiently as possible. The ways in which you can

arrange terminals inside Terminator are the most flexible of the bunch. And if you stop liking a layout, you can just drag and drop each sub-window as you please. The Group Broadcast feature does look a bit like those totally unrealistic Hollywood mockups of computer interfaces, but nobody's forcing you to use it.



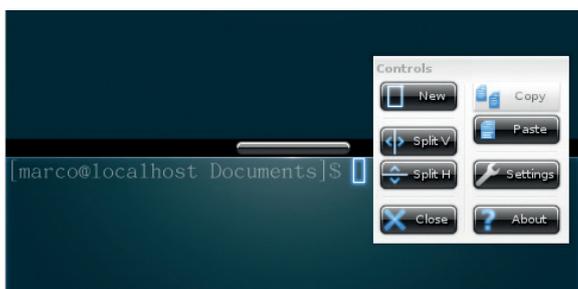
Besides organising terminals in tabs and sub-windows, Terminator makes it easy to control whole groups of them, in broadcast mode.

Terminology ★★★★★

Terminology can use video clips as background, has a themable visual bell (the window flash when it wants to tell you something), and it can display some graphics formats inline.

Every minute spent using this terminal reminds you that there is a whole desktop environment pretty different from the

usual ones, but also quite efficient and interesting. Besides looks, you get splittable tabs and selection of rectangular blocks of texts (a godsend for bug reporting and text processing). Email addresses, URLs and file paths are detected automatically, so copying and pasting them anywhere is a snap.



No menus, no buttons, nothing in sight: Terminology couldn't look more spartan, but its snappy control panel is just one click away.

Flexibility

Bend them to your will!

Terminal emulators can do lots of things that you may not expect of them, or be used in environments different from your everyday GNU/Linux distribution. By definition, this is a sector in which one is forced to compare apples with oranges, so please don't take the corresponding rating too seriously.

Eterm is at its best inside the Enlightenment window manager. Its "auto mode", for example, gets the images to use for scrollbars directly from Enlightenment, and lets the Window manager draw them. At the same time, if you compile and install Escreen code yourself (see the Eterm man page for details), Eterm will get something that is trickier to achieve in its competitors: an interface between the emulator itself and the screen program, to manage multiple (local or remote) terminal sessions via Eterm buttons and menus.

Being part of Enlightenment can take Eterm to places that may be much more difficult to reach for the other emulators. Enlightenment is much more popular for embedded Linux applications than the environments for which the other terminals were designed (with the possible exception of Konsole, if used inside Plasma Active systems). Of course, what we just said for Eterm also applies to Terminology, which besides X11 also works in Wayland and in the basic Linux framebuffer. Other features of Terminology that, in certain scenarios, may be useful to both beginners and power users are its ability to display the content of links inline and to smoothly reflow text when the window is resized.

Other users prefer the Konsole functions to print the content of the window as it was before the execution of the last command, or to save it in plain text or HTML format.

Speaking of flexibility, here is one last thing you may like to know: in KDE, and likely in other environments as well, you can tell any independent window to become a tab of another window. This makes it possible to run all your favourite terminals (each with its own tabs, sub-windows, etc) in just one window.

VERDICT

Eterm	★★★★★
Guake	★★★★★
Konsole	★★★★★
Terminator	★★★★★
Terminology	★★★★★

Looks

Which of these applications is the tallest man in Liliput?

All the emulators we tested can have graphic, more or less transparent backgrounds. Now, if we didn't think that looks matter, we wouldn't have devoted a big box to them, so do yourself a favour and don't waste much time with fancy backgrounds. Back in the 90s, when KDE and GNOME were born, such backgrounds were cool, maybe even necessary, to prove that Linux was not necessarily dull. Today, they just make text harder to read.

The Eterm defaults are the worst from this point of view. Every new Eterm pops up with a different background image, which often seems chosen just to assault your eyes. Besides, the Eterm menus only give three choices of font size (normal, smaller, bigger) and too many choices for brightness, contrast and gamma. Luckily, you can force Eterm to always look exactly as you want with command line switches.

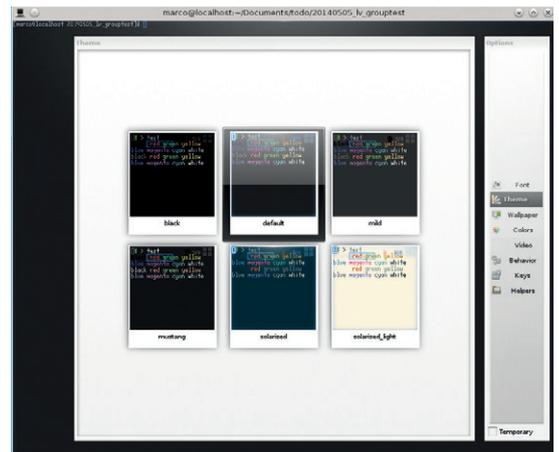
That said, if you do like image backgrounds for text, try the Eterm "viewport" mode: it sets as background,

only for your Eterm instances, an image as big as your desktop, but different from the wallpaper. The result, as the Eterm man page puts it, is "especially keen" if you open several Eterms with the same viewport.

Why be boring?

The Terminology mantra is "Why Should Terminals be boring?" In our opinion, the configuration panels of Terminology are the best-looking of the group. Setting fonts and font sizes, a critical feature, is easy and fun, even if there are many options. Terminology also has a cursor that flashes along as you type, but it manages to do it without being irritating.

There is not much to say about the other competitors, but don't take that as a critique. The graphical configuration interfaces of Terminator and Konsole seem to us the best compromise between ease of use and number of options. Both emulators let you zoom in or out, that is, set different font size in each tab or sub-window, by just pressing Ctrl+ or Ctrl-. In



Testing fonts, backgrounds and themes in Terminology is so easy and fun that you may forget to do any work.

Terminator, even the thickness of the border between terminals in the same window is configurable.

Guake has all the important visual configuration options, and it offers the ability to make scrolling happen either when text output has filled the screen, or only when you hit some key (which is the default).

VERDICT

Eterm	★ ★ ★ ★ ★
Guake	★ ★ ★ ★ ★
Konsole	★ ★ ★ ★ ★
Terminator	★ ★ ★ ★ ★
Terminology	★ ★ ★ ★ ★

Documentation

What should I know, and where can I find it?

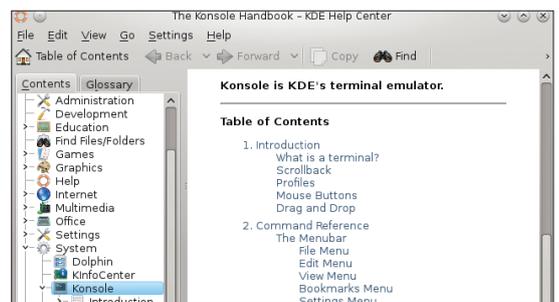
Documentation is a tricky issue, as far as terminal emulators are concerned. That's why it deserves its own section in these pages. The issue we are talking about is related to the very nature of terminal emulators. The right choice of terminal make a big difference in how productive you can be. However, by definition, any program of this kind is just like a connector to the tools that hold the real power: shells and other command interpreters. In other words, how much you can do with an emulator depends first and foremost from how familiar you are with those other programs.

This said, let's look at what the programs of this group test offer when it comes to documentation. Guake seems to have nothing but a very terse

man page. At the opposite end of the scale we have Konsole: in addition to a handy "What's this?" function, one click in its Help menu opens the Konsole section of the KDE handbook. There you'll find almost everything you need to know about this terminal in simple language, including an introduction to how to make scripts interact with Konsole via Dbus.

Terminator only has two man pages: one for the configuration options (**man terminator_config**) and one for the command line switches. The interface, however, is simple enough that this is seldom a problem.

A click on the question mark in the bar of an Eterm window opens the man page of this terminal, in another Eterm window. Other useful, if not mandatory



The Konsole handbook, reachable through the help menu, is clear and complete.

reading, sources of documentation for Eterm are its FAQ and Technical Reference at www.eterm.org. The latter document is the only one that explains how to configure settings such as escape sequences.

VERDICT

Eterm	★ ★ ★ ★ ★
Guake	★ ★ ★ ★ ★
Konsole	★ ★ ★ ★ ★
Terminator	★ ★ ★ ★ ★
Terminology	★ ★ ★ ★ ★

OUR VERDICT

Terminator

If you use the command line just to launch an occasional script a few times a week, practically any terminal emulator would do. As a matter of fact, you should probably attach that script to an icon or menu entry of your desktop, rather than firing up a terminal just to type its name.

In all other cases, which on Linux means “basically always, if you want to fully exploit the power of this operating system” things are much different.

Having several terminals

old computers, where every CPU cycle counts. In all other cases, it has little that you can't find in the others, in a more usable package.

Terminology is perfect for people who want many terminals but get bored by always looking at the same stuff: its configuration panel makes experimenting with colours, fonts and graphic styles so easy that it may kill your productivity.

Konsole (as you'd expect from a KDE application) demands some work to make it behave just as you'd like, but it is powerful and well

“Terminator is not a fashion icon, but is good looking, fast, and above all flexible.”

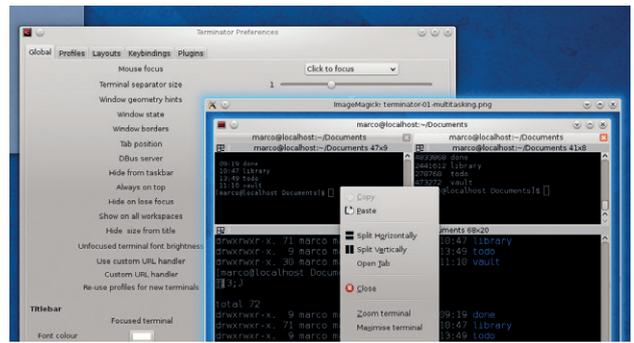
regularly open, each for one different task (eg system monitoring, file searches, web server administration) is not something that only top-notch IT professionals must do – it's for everyone. That's where a terminal emulator that can bundle, preconfigure and run all those shells, makes a big difference.

If you are in the “occasional script” category, go for Guake: besides being made to order for that use case, it looks good and still has tabs and other useful features.

Eterm may be the best option when you need lots of terminals on

pre-configured enough that most people will be happy with it by just opening more tabs.

Terminator is not a fashion icon, but is good looking, at least as fast as the others (on medium-powered computers, at least) and, above all, flexible! If the terminal emulator of your dreams, the one that would make you work happy, looks like a patchwork quilt, be assured that Terminator can look like that, with relatively little effort. This, plus custom commands and other things we already mentioned, make of Terminator the winner of this Group Test 



Terminator: easy configuration interface, endless possibilities.

1st Terminator

Licence GPLv2 Version 0.97

<http://gnometerminator.blogspot.com/p/introduction.html>
Terminator is fast, full of features, has the most flexible layouts and is easy to configure. It also looks good without any fiddling.

2nd Konsole

Licence GPLv2 and GFDL Version 4.12.4

<http://konsole.kde.org/>
The official KDE terminal, perfectly integrated with that desktop, but great even on its own. Not flashy, but solid and complete.

3rd Terminology

Licence BSD Version 0.4.0

www.enlightenment.org
If you need power, but can't do without an original look and feel, Terminology makes it easy to get just what you want.

4th Eterm

Licence BSD Version 0.9.6

www.eterm.org
Not needed in Enlightenment, now that we have Terminology, but it's still powerful and fast.

5th Guake

Licence GPLv2+ Version 0.4.4

www.guake.org
Guake ca,e last simply because it is more specialised than the others, but is a great tool all the same.

YOU MAY ALSO WISH TO TRY...

The terminals presented here are, in our opinion, among the most interesting ones that are available today on Linux. This doesn't mean you should ignore the others, of course. Use this Group Test as a guide to how to evaluate them, instead. Among traditional terminals, for example, you may try lightweight apps like `lxterminal` or `wterm`.

There are many more projects that, while not really ready for prime time yet, may make things much more interesting in a few months. Some are terminals that run inside any browser, like `Anyterm`, `AjaxTerm` or `Shell In A Box`. Then there is `TermKit`, which is a desktop application, but running inside the same WebKit rendering engine used by

`Google Chrome`. Finally, we're keeping an eye on `Final Term`. This aims to redefine the nature and task of a terminal emulator, with features like smart command completion, and above all semantic text menus, which recognise what the piece of terminal output you have selected is, and only display actions compatible with it. Stay tuned!

SUBSCRIBE

shop.linuxvoice.com

Not all Linux magazines are the same



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 - £55
- ROW - £60

DIGITAL SUBSCRIPTION ONLY £38

Each month **Linux Voice** includes 114 pages of in-depth tutorials, features, interviews and reviews, all written by the most experienced journalists in the business.

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.

CLOUDADMIN

System administration technologies brought to you from the coalface of Linux.



Jonathan Roberts dropped out of an MA in Theology to work with Linux. A Fedora advocate and systems administrator, we hear his calming tones whenever we're stuck with something hard.

I've said it before and I'll say it again: as a system administrator, all I want is a quiet life. I want to be able to walk out of work, on call, and know that I'm not going to be woken in the night. Unfortunately, that's just not going to happen. Accidents happen and hardware fails. It's a fact of life, and a perfectly quiet life is not realistic. If not a quiet life, then, we ought at least to aim for an anxiety free life.

Although it passed me by at the time, I now see that 31 March was World Backup Day – a friendly reminder that we should all be backing up our stuff. As a system administrator, just knowing that you're backing up all your important data, checking that the backups are actually running and that you can restore them in case of disaster, will go a long way to reducing your anxiety. In an age of configuration management, Amazon Web Services and the like, you should go one step further: you ought to know that you can quickly, accurately and automatically rebuild every single one of your systems.

Destroy to rebuild

Ideally, this means taking advantage of all that redundancy you've built in to your platform. Running a web farm behind a load balancer? When you release new software, don't just push it out to the existing servers; build new servers and destroy the old ones. Running a database cluster and doing an OS upgrade? Don't just do a yum update and reboot; rebuild the server, restore your data and destroy the old one.

Having done this a thousand times, knowing that you can rebuild your systems and restore critical data to them will cut out so much anxiety from your life as a system administrator.

Logs

Get started with your distro's built-in log management tools.

After three months looking at exciting new technologies, this issue I want to step back and look at something far less glamorous: logging. Often an afterthought to developers and system administrators alike, collecting, monitoring and using logs is one of the best things you can do for your infrastructure. Let's start with the basics – what are logs and why should you care?

Logs are how non-interactive components of your system communicate with you, the user. Each time an action takes place a note will get written to a file (the 'log') somewhere on your hard drive. Most logs contain one entry per line in the file. In the case of a web server, for example, an entry will probably be in the Common Log Format:

```
192.168.79.2 - - [27/Apr/2014:13:55:36 +0000] "GET /lv.html HTTP/1.1" 200 2326
```

Here, you see:

- The IP address of the user who visited the website.
- (Two empty fields not logged in this example, represented by dashes).
- The date time and timezone when the server finished dealing with the request.

- The request itself.
- The status code (the result of the request).
- How many bytes were returned to the clients.

It only takes a little bit of imagination to see how storing all this information can be useful. You can map IP addresses to geographic location, letting you know where in the world your site's visitors are coming from; you can see which web pages are most popular; you can even see if any pages experience more errors than others and at what times those errors occur.

Other logs on your system can help you keep things secure, too. For instance, on Red Hat-based systems, you can look in the `/var/log/secure` file and see a record of who logged in and when. You will also be able to see failed login attempts, commands executed with sudo and more besides. In the event of a security incident, your logs can be an invaluable source of information to help you understand how it happened, what damage was done, and how to stop it happening again in the future.

Now you know what logs are and why you should care about them, let's look at how you

The screenshot shows a web browser window titled "rsyslog documentation - Mozilla Firefox". The address bar shows the file path `file:///usr/share/doc/rsyslog-doc/manual.html`. The page content includes the title "RSyslog - Documentation" and a detailed introduction to the software, its features, and how to use it. It also mentions project sponsors and provides links to various resources like troubleshooting, configuration, and bug lists.

RSyslog comes with a comprehensive set of documentation if you install the `rsyslog-doc` package. You'll find it installed in `/usr/share/doc/rsyslog-docs/`.

manage them. While not quite universal, you'll find many of your system's logs (on Red Hat based systems, at least) are managed by a program called RSyslog – the 'rocket-fast system for log processing'. It implements the syslog standard, enabling you to store logs locally or ship them remotely and store them in different formats.

RSyslog

In syslog-based systems, each message gets categorised with a facility and a priority. The facility describes which subsystem generated the messages, and can be one of auth, authpriv, cron, daemon, kern, lpr, mail, news, syslog, user, uucp, and local0 through local7. The local0–7 facilities are used for custom applications and reserved for the user to specify.

The priority describes whether the message is just informational or whether it relates to one of various error states. Available priorities are debug, info, notice, warning, err, crit, alert, and emerg.

The syslog daemon your computer is running will be configured to store messages of different facility and priority combinations in different locations. On a Red Hat-based system, you can see what the default configuration is like by looking in **/etc/rsyslog.conf**. For now, just skip to the **#### RULES ####** section.

Here, you can see lines defining where different messages go. To explain the syntax, look at this example:

```
*.info /var/log/messages
```

On the left is a filter to match log messages: in this example, messages of any facility with the priority **info** will be matched. On the right is the action – that is, what should happen to the message. In this example, the matched messages will be sent to the file **/var/log/messages** (note that most log messages will be stored in the **/var/log** directory by default – it's always a good place to start when looking for a log).

If you want to, you can use this file to redirect certain log messages to different files. Be sure to restart the **rsyslog** daemon after you make changes. Remember that many application might define their own logging arrangements, so you could find, for example, that your web server has its own log configuration in **/etc/httpd/conf/httpd.conf** or in individual virtual host definitions. Keep this in mind if you can't find the log file you're looking for.

If you want to interact with the syslog implementation from shell scripts you've

```

Apr 29 17:23:02 localhost systemd: Started dnsmasq.
Apr 29 17:28:37 localhost dnsmasq[1520]: DHCPREQUEST on wlp2s0 to 192.168.1.254 port 67 (xid=0x39e23547)
Apr 29 17:28:37 localhost dnsmasq[1520]: DHCPACK from 192.168.1.254 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> (wlp2s0): DHCPv4 state changed -> renew
Apr 29 17:28:37 localhost dnsmasq[1520]: bound to 192.168.1.83 -- renewal in 38023 seconds.
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> address 192.168.1.83
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> plen 24 (255.255.255.0)
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> gateway 192.168.1.254
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> server identifier 192.168.1.254
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> lease time 86400
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> nameserver '192.168.1.254'
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> domain name 'lan'
Apr 29 17:28:37 localhost dbus-daemon[917]: [system] Activating via systemd: service name='org.freedesktop.nm_dispatcher' unit='dbus-org.freedesktop.nm_dispatcher.service'
Apr 29 17:28:37 localhost dbus-daemon[917]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'
Apr 29 17:28:37 localhost NetworkManager: DHCPREQUEST on wlp2s0 to 192.168.1.254 port 67 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager: DHCPACK from 192.168.1.254 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager: bound to 192.168.1.83 -- renewal in 38023 seconds.
Apr 29 17:28:37 localhost dbus-daemon[917]: [system] Activating via systemd: service name='org.freedesktop.nm_dispatcher' unit='dbus-org.freedesktop.nm_dispatcher.service'
Apr 29 17:28:37 localhost dbus-daemon[917]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'
Apr 29 17:28:37 localhost systemd: Starting Network Manager Script Dispatcher Service...
Apr 29 17:28:37 localhost systemd: Started Network Manager Script Dispatcher Service.
Apr 29 17:40:19 localhost dbus-daemon[917]: [system] Activating via systemd: service name='net.reactivated.Fprint' unit='fprintd.service'
Apr 29 17:40:19 localhost dbus-daemon[917]: [system] Successfully activated service 'net.reactivated.Fprint'
Apr 29 17:40:19 localhost systemd: Starting Fingerprint Authentication Daemon...
Apr 29 17:40:19 localhost systemd: Started Fingerprint Authentication Daemon.
Apr 29 17:40:19 localhost fprintd: Launching FprintObject
    
```

/var/log/messages is overwhelming at first, but it contains a treasure trove of information. Next time you're trying to figure out why something doesn't work, make **/var/log** your first stop.

written (backup scripts etc), then you can use the **logger** command. Use is simple:

```
logger -p local2.info "Database backup started."
```

This will send your defined message to syslog with the facility **local2** and priority **info**. You can then configure where such messages end up by adding a new entry to the **/etc/rsyslog.conf** file.

What's great about using logger is that you don't have to worry about formatting your message: logger will automatically append date, host and user information to your messages.

Try playing with this to see if you can generate a message in **/var/log/messages** – on most systems, if you run logger without the **-p** option, you'll find your message in **/var/log/messages** due to a line similar to the following in **/etc/rsyslog.conf**:

```
*.info;mail.none;authpriv.none;cron.none /var/log/messages
```

Logrotate

As great as syslog is, if you ever find yourself managing a cluster of more than two servers, you'll quickly find yourself getting annoyed with logging in to many different servers just to get a picture of what's happening across your entire cluster.

Thankfully, RSyslog provides the means by which you can centralise your logs. For example:

```
**. @192.168.79.10
```

Putting this line in your RSyslog configuration file will instruct RSyslog to forward messages of all facilities and priorities to the machine at 192.168.79.10. Pretty simple, and you can filter messages in exactly the same way as shown for local logging, too. Setting up the receiving server isn't much more difficult:

```

$ModLoad imudp
$UDPServerRun 514
...
$template web,"/var/log/remote/%fromhost%/%year%/%month%/%day%/%error.log"
...
local7.error ?web
    
```

This snippet instructs RSyslog to load the **imudp** module, which lets it receive logs via the UDP protocol, and instructs it to listen on port 514. Below that, we define a template for the output file name. In this example, we've included a number of properties that will get expanded by RSyslog on receipt of messages to create the full path to the log file. Finally, we filter all **local7** facility error messages to go to this template (the **?** indicates that a template is being used in the action).

Remotely storing logs like this isn't just convenient, it can help improve security, too. In the event of a breach, you have a second set of logs kept elsewhere. If the attacker wants to cover their tracks, they now have to break into a further server if they're to get at all copies of the logs.

What next?

That was a brief introduction to logging in Linux and rsyslog in particular. There's much more you can do with logs, however, and next month's issue we'll introduce you to Logstash. This open source tool can be taught how to parse your logs, after which it will index them, make it easy for you to generate pretty graphs and extract useful information from all those text files – all without having to touch grep and with lightning speed.

Until then, run back to your desk and sort out your logging – you'll thank us!



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

CORE TECHNOLOGY

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

Processes

All the world's a Unix filesystem/and all the processes merely players...

You're undoubtedly aware that Linux, like most modern operating systems, can run many applications at the same time, through a feature called multi-processing, or multi-tasking. I can easily count the number of processes that my machine is running with a command such as:

```
ps -e | wc -l
```

The answer turns out to be 175, which seems quite a lot for a humble laptop that's basically just editing a text file.

So how do all these processes come into being? Well, some of them are started at boot time. These are the system services (usually referred to as "daemons"), and many of them may continue to run indefinitely, until the system is shut down. Some services (such as the Apache web server) maintain a pool of processes to enable them to promptly service multiple concurrent clients. Quite a lot of processes come into existence when the desktop is started. Of the 175 processes I counted earlier, rather more than 60 are there to give life to my desktop or to run the applications that are active there. Finally, some processes are started by my own actions when I launch an application from a menu or dashboard, or I type a command at a shell prompt.

A process is sometimes defined as "an instance of a program in execution". The analogy of an actor reading from the script of a play may be useful – the script represents the program: the passive list of instructions of what's to be done. The actor represents the process, the active entity responsible for carrying out those instructions. But perhaps the best way to get a handle on what a process is is to delve more deeply into the resources and

information it carries. Take a look at the table, below.

Everything you know is wrong

Even if you're not a programmer, you'll be used to some of these concepts. Take the notion of a current directory for instance (as reported by the **pwd** command). You think of yourself as being "in" a particular directory. But really, it's the process that's running your shell that holds the concept of current directory. You're probably also aware that you're running as a specific user, as reported by the **id** command. But again, it's really the process that's running your shell that maintains the notion of user identity. This is the identity that's used when access control checks are made – am I allowed to write to such-and-such a file, for example.

Although there are many circumstances under which a new process is started, there

is only one way to actually do it, and that's for a process to create a copy of itself. This operation is called forking. The new process is called the child and the original process is called the parent. The child shares the code segment with the parent... that is, it is executing the same program. It receives a copy of the data segment, the stack, and the heap. (The reality is actually a bit more complicated. These memory regions are not copied piecemeal, but the "copy-on-write" operation of the virtual memory system makes things behave as if they were.) The child also inherits many other things – user ID, current directory and so on – from the parent. In fact, the child begins life essentially as an exact clone of the parent except for one small thing – it has a different process ID. Returning to our actor/script analogy, forking is roughly analogous to our original (parent) actor calling out to the

Information carried by a process	
Item	Description
Process ID (PID)	Unique integer identifying the process
Code segment	Memory region that holds the program's executable instructions
Data segment	Memory region that holds statically defined data
Stack	Expandable memory region used to store 'local' function variables
Heap	Expandable memory region used to store dynamically allocated objects
Priority	Determines how favoured the process will be by the scheduler. Derived from the 'nice' value
Signal disposition	The way that the process responds to the receipts of signals – are they caught, ignored, etc.
Environment	A list of environment variables of the form NAME=VALUE
File descriptors	'Handles' on streams that are available for reading or writing (includes stdin, stdout and stderr)
UID	Real user identity
EUID	Effective user identity
CWD	Current working directory

Try It Out

Compile C code

To compile the C examples in this tutorial you'll need to install the C compiler (GCC). On a Debian-style system use this command:

```
# apt-get install gcc
```

and on a Red Hat-style system:

```
# yum install gcc
```

Now create a file called `forkdemo.c` with content as follows:

```
#include <stdio.h>
void main()
{
    int i;
    if (fork()) {
        for (i=0; i<10000; i++)
            printf("*** PARENT %d\n", i);
    }
}
```

```
} else {
    for (i=0; i<10000; i++)
        printf("**CHILD %d\n", i);
}
}
```

To compile the program, run the command:

```
$ gcc -o forkdemo forkdemo.c
```

and assuming you don't get any compilation errors, run it like this...

```
$ ./forkdemo
```

The thing to notice here is that the output from the parent and the child is interleaved in a non-deterministic way, as the scheduler switches back and forth between the processes. It's also possible (rather likely in fact) that some of the CHILD messages will appear AFTER the shell's next prompt.

wings for a new actor, who trots out onto the stage and stands by the side of the first actor, sharing a copy of the script.

Scary C code

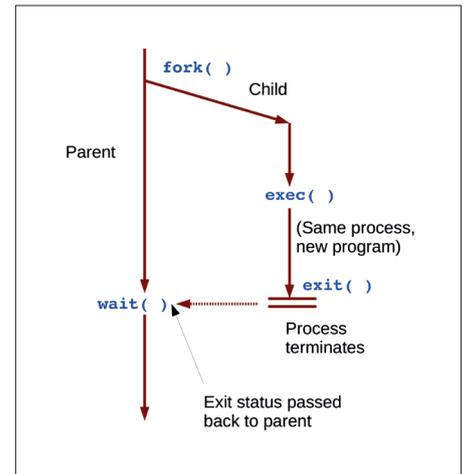
To see how this really works I'm going to inflict a few lines of C code on you. The action here centres around the `fork()` system call. It's a simple enough call – it takes no arguments and returns just an integer result – but of all the system calls in Linux it's perhaps the one that is hardest to get to grips with, because although only one process makes the call, two processes return from it – the original parent and the new child. They figure out which is which by examining the return value from the call. In the parent, `fork()` returns the PID of the newly-created child. In the child, it returns zero. So you invariably see the `fork()` call wrapped inside an `if` test. Keeping in mind that in C any non-zero integer value counts as "true", and zero counts as "false", the code might look something like the boxout above.

Sometimes, parent and child continue to execute the same program. This happens, for example, when Apache spawns its pool of "spare" server processes. More often, though, the child will start to execute a different program. This is what happens, for example, when I type a command into the shell. (I'm talking about running a command that lives in an external executable file, not a built-in shell command such as `echo` or `cd`.) A process runs a new program by executing an `exec()` system call. In doing this, the code, date, stack and heap of the old program are discarded; however, the process retains its process ID, its current directory, its open file descriptors, and (usually) its environment. To return to our actor/script analogy, an

`exec()` is akin to the actor discarding his current script, picking up a copy of Macbeth, and starting at the beginning. He remains the same actor, but he's performing a different play. There are, in fact, six versions of the `exec` call, but I will not burden you with the minutiae of their differences. The call is unusual in that it doesn't return except in the case that it fails (usually because it can't find the executable).

Two other system calls are important for management of processes – `exit()` and `wait()`. The `exit()` call is simple enough – it terminates the process. You can pass an integer argument to `exit()`, which is passed back to the parent process and is known as the "exit status". The convention is that an exit status of zero indicates a normal, successful termination, and non-zero values indicate failure of some sort.

For programs started from a shell prompt, you can find the exit status of the



The classic fork/exec/exit/wait model. This is what happens every time you run a command.

last command through the variable `$?` . For example,

```
$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
$ echo $?
0
```

```
$ grep xyzzy /etc/passwd
```

```
$ echo $?
1
```

```
$ grep root /etc/xyzzy
grep: /etc/xyzzy: No such file or directory
```

```
$ echo $?
2
```

Here we see that `grep` returns an exit status of 1 if it doesn't find the pattern and 2 if it can't find the input file.

The `wait()` call is used by a parent process to wait until its child process (or one of its children) terminates. This call can also be used to retrieve the exit status of the child.

Using `fork()`, `exec()`, `exit()` and `wait()` we can write a minimal shell:

Try It Out

Write a shell

Create a file called `tinyshell.c` with this content:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
/* A minimal shell */
main()
{
    char line[100];
    /* Main command loop */
    while (printf("> ", gets(line)) != NULL) {
        if (fork() == 0) { /* Child */
            execlp(line, line, (char *)0);
            /* Don't come here unless execlp fails */
            printf("%s: not found\n", line);
            exit(1);
        }
    }
}
```

```
}
else wait(0); /* Parent */
}
}
```

Now compile and run it something like this:

```
$ gcc -o tinyshell tinyshell.c
```

```
$ ./tinyshell
```

```
> date
```

```
Wed Mar 19 15:25:35 GMT 2014
```

```
> hostname
```

```
ubuntu1204
```

```
> who
```

```
chris tty7 2014-03-19 13:52
```

```
chris pts/0 2014-03-19 14:05 (:0)
```

```
> ls -l
```

```
ls -l: not found
```

Try It Out

Braaaaaains!

It's rather easy to make zombies. Create a file called `makezombies.c` with content as follows:

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int i;
    for (i=0; i<5; i++)
        if (fork() == 0)
            exit(0); /* Child */
    sleep(1000);
}
```

Then compile it and run it in the background:

```
$ gcc -o makezombies makezombies.c
```

```
$ ./makezombies &
```

Now if you run `ps` you'll see a parent and five zombie children:

```
$ ps
PID TTY    TIME CMD
5982 pts/0  00:00:00 makezombies
```

```
5983 pts/0  00:00:00 makezombies <defunct>
5984 pts/0  00:00:00 makezombies <defunct>
5985 pts/0  00:00:00 makezombies <defunct>
5986 pts/0  00:00:00 makezombies <defunct>
5987 pts/0  00:00:00 makezombies <defunct>
5988 pts/0  00:00:00 ps
32052 pts/0  00:00:00 bash
```

The point of this example is that the parent creates five children that immediately exit. Rather than waiting for them, the parent enters a 1000-second sleep. You cannot kill the zombies in the usual way. For example, the command

```
$ kill 5987
```

has no effect.

The trick to getting rid of zombies is to track down and kill the parent, which is easy to find in this case:

```
$ kill 5982
```

When the parent dies, the zombie children are inherited by `init` (PID=1) which will collect the exit status from each of them, allowing them to be finally laid to rest.

You'll notice that the last example fails – the shell doesn't understand about command options and tries to find a file called `ls -l`. Well, honestly, what do you expect from 12 lines of code? But despite its simplicity, this little program illustrates the essence of what happens when I run a command in the shell.

With this model in mind, it's easy to see how the shell implements background jobs (when the command line ends with `&`). The shell simply doesn't do the `wait()` before it prompts for the next command.

Occasionally, things go wrong with this model. If a process exits, and its parent is not waiting for it, the child enters a "defunct" state (also known as a "zombie"). The child cannot be completely laid to rest because it needs a waiting parent to pass its exit status back to. Here's an analogy. Little Jimmy, who's only six, is usually met by his mother after school. She waits for him by the school gate and Jimmy is always excited to see her. One day, mum gets held up in traffic and isn't there when Johnny comes out of school. He has no-one to report his exit status to, and enters a zombie state. (Of course, analogies should not be pushed too far!)

Let's move away from C code and take a look at processes from the command line. The classic command for viewing processes is `ps`. Other commands you might find helpful are `pstree`, which shows the process ancestry using simple ASCII art, and `pgrep`, which reports the process ID(s) of the processes that are running a specified

program, or have a specified owner or a specified parent, and so on.

A figment of the kernel's imagination

Commands such as `ps` garner most of their information from the `proc` filesystem, which is usually mounted on the `/proc` directory. Through this filesystem the kernel exposes a large amount of process information in the form of what it calls 'files'. They are not files as we normally think of them; they are an illusion provided by the kernel to give us a file-like view of some of its internal data structures. You can tell there's something funny going on because the command

```
$ ls -l /proc
```

shows most of the 'files' as having zero length, but if you read from them there is content. Try this:

```
$ cat /proc/partitions
```

for example. A directory listing of `/proc` will reveal a number of subdirectories named directly after the process IDs. In these

subdirectories you'll see a standard set of 'files' that expose per-process information. Let's change to the directory corresponding to the process running my `rsyslog` daemon:

```
$ pgrep rsyslog
527
$ cd /proc/527
```

Much of this information you'll find here is very low-level. Try this:

```
$ cat status
```

You'll see lots of memory usage information and stuff about signal dispositions that is downright inscrutable. Usually, the output from programs such as `ps` and `lsdf` will be easier to scute (is that a word?). Let's delve a little deeper though. The subdirectory "fd" contains symbolic links that are named after the process's open file descriptors:

```
$ sudo ls -l fd
total 0
lrwx----- 1 root root 64 Mar 21 14:40 0 ->
socket:[7307]
l-wx----- 1 root root 64 Mar 21 14:40 1 -> /var/log/
syslog
l-wx----- 1 root root 64 Mar 21 14:40 2 -> /var/log/
mail.log
lrwx----- 1 root root 64 Mar 21 14:40 3 ->
socket:[7309]
lr-x----- 1 root root 64 Mar 21 14:40 4 -> /proc/
kmsg
l-wx----- 1 root root 64 Mar 21 14:40 5 -> /var/log/
auth.log
l-wx----- 1 root root 64 Mar 21 14:40 6 -> /var/log/
kern.log
```

We see here that file descriptor 1 (for example) is connected to `/var/log/syslog`.

The environment

One important piece of baggage that a process carries around with it is the environment, which is basically just a list of strings, each of the form `NAME=VALUE`. A child process inherits its environment from its parent after a `fork()` and the environment usually remains intact across an `exec()`, too, depending on which version of the `exec()` call is used. From your command prompt

Threads

Linux allows multiple concurrent paths of execution within the address space of a single process. Some people refer to threads as "lightweight processes" because they carry a lot less context around with them than regular processes, and are much cheaper to create. Threads exist within a process; they rely on the process to carry around most of the execution context. By default a process has only one execution thread. In a more thread-oriented view of the world, you could consider the process as

being the passive holder of context, and the thread as being the active entity that gets stuff done.

Here's an analogy. Sue (the process) takes her three children (the threads) on a picnic. She struggles up the hill to a nice spot overlooking the river carrying a blanket, a food hamper, and some folding chairs. She sets them all out on the ground. The kids, meanwhile, just have a good time running around. They are not laden with hampers or blankets – they carry with them almost no context of their own.

you can examine the environment of your shell's process with the **env** command:

```
$ env
SSH_AGENT_PID=31487
TERM=xterm
SHELL=/bin/bash
USER=chris
LANG=en_GB.UTF-8
HOME=/home/chris
```

The list I've shown here is vastly stripped down. As they are inherited, environment variables can be used to pass configuration information to applications. Examples include **EDITOR**, (which tells commands

like **crontab** and the command line mail program which editor to use) **DISPLAY** (which tells graphical programs where their X server is) and **CLASSPATH**, which tells Java apps where to look for their class files. Environment variables are commonly set in shell startup files such as **/etc/profile**.

As a final thought

There's quite a bit of interest at the moment in Linux Containers, which is finally reaching some level of maturity. In his excellent feature about them in LV002, Jonathan Roberts wrote: "The idea of containers is to

make it easy to run multiple applications on a single host, all the while ensuring each remains separate."

But you can do that with processes too! In fact we can see threads and processes as one end of a spectrum of containment technologies, with chroot jails, containers and full-blown virtualisation at the other end. As we move across the spectrum, less is shared and more is separate. For example, processes all share the same TCP/IP port space – I can't have two processes binding to port 22. Each container, on the other hand, provides a separate port space. 📺

Command of the month: ps

ps is the classic command for listing the processes running on the machine. Used without options, it gives you a minimum of information about the processes associated with your terminal:

```
$ ps
PID TTY          TIME CMD
352 pts/0        00:00:00 ps
32052 pts/0      00:00:00 bash
```

Typically, this output will show only the **ps** command itself, and its parent shell (**bash**). How come the PID for the **ps** command (352) is less than the PID of its parent (32052)? That's because after PIDs have reached a specified limit (by default it's 32768) the numbers wrap around and the PIDs get recycled. I didn't deliberately engineer that result, it just happened to come out that way.

ps has a large and very confusing set of options, partly because it's trying to remain backwards compatible with both of its progenitors (the BSD and System V versions) and partly because, well, it's just got a lot of options. Some of them control which processes are listed; some control how much detail is shown. I suspect that most people just remember two or three useful, and stick with those.

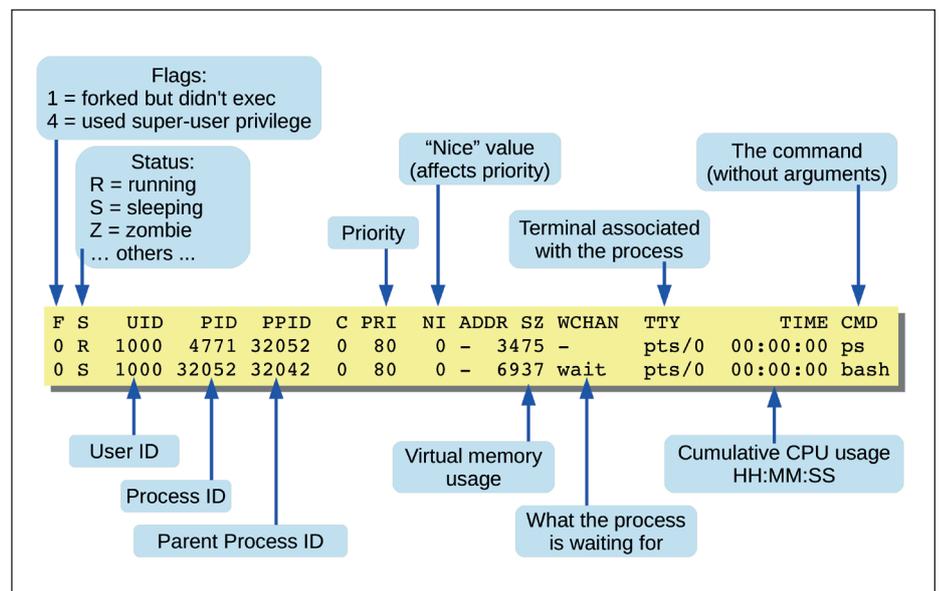
ps in depth

The **-f** option gives a full listing:

```
$ ps -f
UID      PID  PPID  C  STIME TTY          TIME CMD
chris    4770 32052  0  12:33 pts/0        00:00:00 ps -f
chris    32052 32042  0  Mar19 pts/0        00:00:00 bash
```

...and the **-l** option gives even more (rather like the **-l** option of **ls**):

```
$ ps -l
```



The **ps** command sure generates a lot of output. But what does it all mean?

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY          TIME CMD
0 R 1000 4771 32052 0 80 0 - 3475 - pts/0
0 S 1000 32052 32042 0 80 0 - 6937 wait pts/0
00:00:00 ps
00:00:00 bash
```

The divided heritage of **ps** causes confusion. For example the two commands

```
$ ps -elf
```

```
$ ps aux
```

produce approximately the same output.

The first uses System V syntax, the second uses BSD syntax (note the absence of a hyphen). They do not, however, show exactly the same output fields. I suspect that most administrators memorise two or three useful combinations of options and stick with those. That's what I do!

If you want detailed control over the output fields use the **-o** option. This example

shows the process ID, group ID, and command line:

```
$ ps -o pid,gid,cmd
PID  GID  CMD
6260 1000 ps -o pid,gid,cmd
32052 1000 bash
```

There are LOTS of output columns you can select here; see the STANDARD FORMAT SPECIFIERS section of the man page for the details.

The **-e** option lists every process, but you can also get finer control over which processes are listed. For example,

```
$ ps --ppid 1
```

shows those processes whose parent is PID 1 (note the use of the GNU-style command option with the double hyphen!), or to show just the processes running as **chris**:

```
$ ps -u chris
```

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...

CD/DVD image editor

ISO Master 1.3.11

If you've ever taken a peek at the **genisoimage** or **mkisofs** manual pages, you'd be forgiven if you come away shuddering. The former is a whopping 1,596 lines long – and that's not a bad thing, because its level of detail is superb. But the number of options included in these command line CD/DVD creating tools is overwhelming, especially if you're not *au fait* with making disc images and you just want to knock together a quick backup DVD.

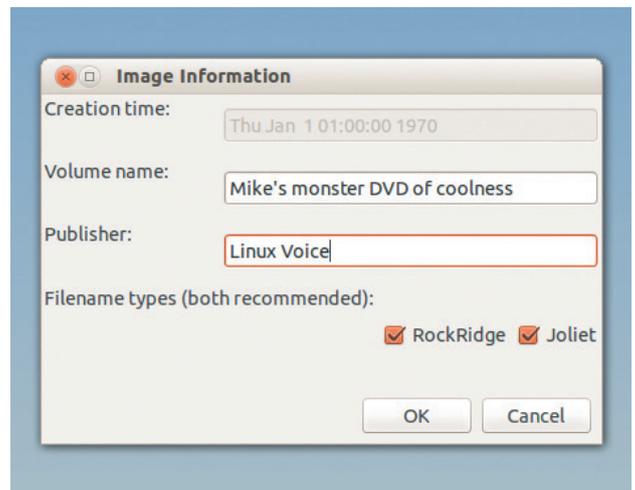
ISO Master aims to be a simpler and friendlier graphical alternative. But it's not just limited to basic data discs; it has options for making bootable discs with multiple filesystem formats as well. GTK 2 is used to provide the interface, so to compile ISO Master from source, you'll need the development packages installed (try **libgtk2.0-dev** under Debian/Ubuntu-based

distros). Extract the **.tar.bz2** file, change into the resulting directory with **cd** and run **make** followed by **sudo make install**. Then start the program with **isomaster**.

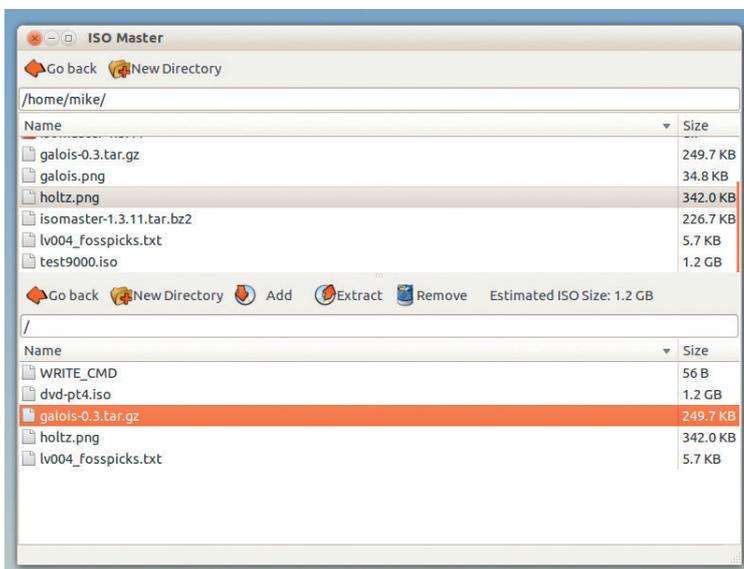
Functional minimalism

Many disc authoring tools clog up the screen with toolbars, panels and directory trees all over the place, but ISO Master has just two panels: the top one browses your filesystem, and the bottom one browses the contents of the disc. You can open an existing ISO image by double-clicking on one in the top panel, or create a new image via the File > New menu. Click a file or directory in the top panel, then Add

“ISO Master aims to be a simpler and friendlier graphical alternative to genisoimage and mkisofs.”



Rock Ridge and Joliet filesystem extensions for CDs/DVDs provide better integration with Unix and Windows systems respectively.



ISO Master shows the estimated size of the resulting ISO on the right-hand size, between the two panes.

to add it to the DVD, or Extract to remove a file from an ISO.

To change filesystem options, go to File > Properties; there you can choose whether Rock Ridge and Joliet extensions should be enabled (a good idea, as these let discs use longer filenames), and you can also enter text for the volume name and publisher. Use the Tools > Boot Record menu to add a boot image to a disc, eg if you're making a bootable Linux distro.

When you're finished creating the disc, click File > Save As and provide a filename. ISO Master will generate **<filename>.iso**, ready to be burned to a CD/DVD-R or booted in a virtual machine such as VirtualBox. It's simple, fast and trouble-free – great work, team!

PROJECT WEBSITE
www.littleavr.ca/isomaster

Social networking client

Dianara 1.2.0

The big social networking sites, such as Facebook, Google+ and Twitter, have two major problems: they're all massively centralised, and they retain scary amounts of personal data about us. It's not all bad though. Pump.io is an example of a social networking service that is both open source and decentralised – that is, anyone can run a server that becomes part of the network. So no single company owns or controls the service, and if one machine goes down, users registered on another machine won't be affected. Anyone can set up their own Pump.io node (even on low-spec kit such as a Raspberry Pi), so if anyone is interested in a tutorial, let us know.

Anyway, Dianara is a graphical application for accessing your Pump.io account without having to go through a web browser. Its interface is built using Qt, and to install it you'll need the **qjson-devel**, **qoauth-devel** and **libmagic-devel** packages. Extract the **dianara-v1.2.0.tar.gz** archive and see the **INSTALL** file inside for detailed installation instructions, including a helpful list of dependencies for many popular distros.

When you first start Dianara, you'll be prompted to link it to your

Pump.io account. And if you don't have one of those yet, you can get one from many different sites – we used <https://pumpit.info>. Reassuringly, the Pump.io service doesn't just let random applications connect to your account and post things (a sure source of spam), so Dianara needs a validation key and confirmation from the web interface that you want to link the program to your account.

Pump.io up the volume

Pump.io is a relatively simple service compared with the heavyweights of Facebook and Google+, and this is reflected in the application. You can post messages to your timeline (plain text, or with media attachments), follow people to see what they're up to, and click stars on posts to add them to your Favourites list. You can also add comments to posts, share them with other people, and set up lists (to separate friends from colleagues, for instance).

In all it's an attractive and newbie-friendly program that still

“Maybe we can finally show that info-hoarding Zuckerberg fella who's boss?”



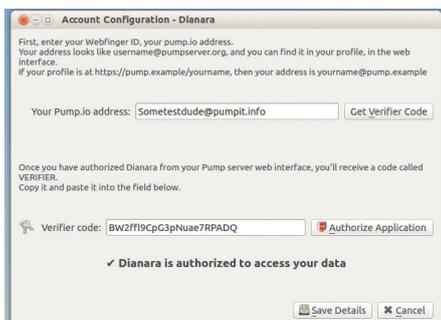
Dianara crams plenty of information and widgets into its window, making better use of space than the web interface.

packs in a good amount of functionality. It's also decently configurable, with options for changing how notifications work and how often the timeline should be updated. There's not much in the way of online help, although the tips are useful.

The biggest problem for Pump.io is its relatively small userbase. But it's a great system, so if we all get on Pump.io, maybe we can finally show that info-hoarding Zuckerberg fella who's boss...

PROJECT WEBSITE
<http://jancoding.wordpress.com/dianara>

How it works: Get posting on Pump.io



1 Sign up

Create an account at <https://pumpit.info>, then start Dianara and link it to your account. (You will be asked to get a verification key from the site.)



2 Add contacts

Click the Contacts tab and add people you know. If you don't know anyone yet, try adding [evan@e14n.com](http://e14n.com), aka Evan Prodromou, Pump.io's creator.



3 Post a message

Click the Timeline tab and click in the text field at the top. Give your post a title, add some text or an image, and click Post to share with your followers.

Process sandboxing tool

Firejail 0.9

Containers, virtual machines, chroot jails... There are many ways to run programs in restricted sandboxes, where they can't interfere with the rest of the system. This is great idea if you're ramping up security measures, or you're running a program from an unknown source and want to make sure it doesn't hose your distro. Setting up these sandboxes can be tricky, though – so Firejail provides an easier alternative.

Both RPM and Deb binary packages are available to download from the program's website, or you can compile it from its source code (it has no unusual dependencies and should install on any recent distro). Once you have it installed, run it along with a program like so:

firejail bash

Now, inside the new Bash shell session, enter **ps ax**. Notice

something strange? Yes, there are only two processes listed – and PID 1 (which is normally **/sbin/init**) is the current Bash process. Firejail has created its own process namespace for this Bash session, so it can't poke around and discover anything else that's running.

Another useful option is:

firejail --overlay bash

If you have OverlayFS enabled in your kernel, this will overlay a temporary filesystem onto the current one. So you can create and modify files inside the jailed environment, but when you access it, all of the changes are forgotten. This is great when you have programs that need to write files, but you want to revert to the original state of the filesystem when you close them.

Firejail includes various other ways to sandbox programs, like

```
mike@mike-megabox: ~
mike@mike-megabox:~$ cat test.txt
Just one line in this file!
mike@mike-megabox:~$ firejail --overlay bash
Parent pid 4695, child pid 4696
Interface      IP                Mask              Status
lo             127.0.0.1        255.0.0.0        UP
eth1          192.168.1.55    255.255.255.0    UP

Child process initialized
[mike@mike-megabox ~]$ ps ax
  PID TTY          STAT       TIME COMMAND
   49 pts/3        S           0:00 bash
  499 pts/3        R+          0:00 ps ax
[mike@mike-megabox ~]$ echo "Another line" >> test.txt
[mike@mike-megabox ~]$ cat test.txt
Just one line in this file!
Another line
[mike@mike-megabox ~]$ exit
mike@mike-megabox:~$ cat test.txt
Just one line in this file!
mike@mike-megabox:~$
```

With the **--overlay** option, file changes inside a Firejail session are forgotten when the session ends.

providing them with their own TCP/IP stacks, and it includes a simple monitoring tool (firemon, which shows activity in Firejail sessions). In all it's a simple and user-friendly way to run applications in restricted environments, so if you've been bamboozled with VMs, containers and chroot commands, give it a go.

PROJECT WEBSITE
<http://l3net.wordpress.com/projects/firejail>

Internet radio browser

Streamtuner2 2.1.0

For various reasons, mostly to do with mountains and wheat beer, this author started learning German four years ago. It has been a tough ride, but one of the things that helped is the abundance of German radio stations on the internet.

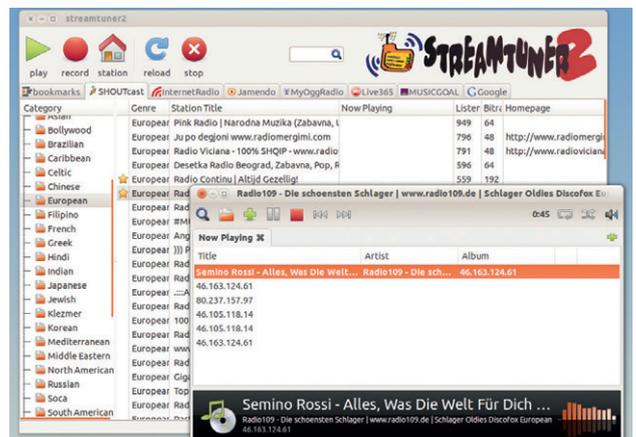
There are thousands of radio stations being broadcast via the internet, and while you can access them via your browser, it's easier and faster to search via a dedicated program. Streamtuner2 is one such app: it's written in Python with GTK as its front-end, so you'll need the **pygtk** package installed to run it. Generic Deb and RPM packages are available on the project's website, and we had no problem installing the Deb on Ubuntu 13.10.

Streamtuner2 doesn't play audio itself – you'll need a standalone

player such as Audacious to hear the streams. When you start Streamtuner2 for the first time, it asks you which player you want to use. Then it presents you with several internet radio station directories, some of which didn't work in our testing, which was slightly annoying. We had the most luck with the SHOUTcast and Music Goal services.

With thousands of stations to browse, across all manner of genres and languages, it can be difficult to remember your favourites. So Streamtuner2 includes a handy bookmarking system: right-click on a station and

“Streamtuner2 includes a handy bookmarking system.”



Fancy some Top Albania Radio? Or perhaps some Bluegrass Jamboree? Streamtuner2 has it all.

choose bookmark. You'll then find that station under the Bookmarks tab, on the far left of the tab bar.

Streamtuner2 has some rough edges in the layout and wording used in the interface, but it's still a good way to explore the vast world of internet radio.

PROJECT WEBSITE
<http://milki.include-once.org/streamtuner2>

Compression utility

PeaZip 5.3

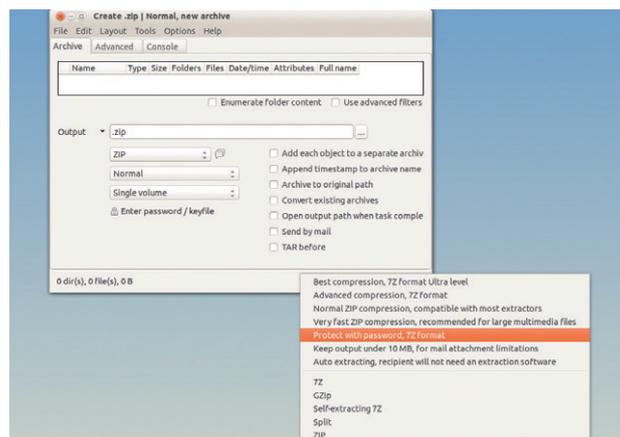
PeaZip is a graphical application for creating and extracting compressed archives, and as of the latest release, it supports over 150 archive formats. (Yes, 150 – that’s not a typo!) Along with the usual suspects such as Zip and **.tar.gz**, the program can work with relatively obscure formats such as **.arc** and **.arj**. If you created some archives with a random piece of shareware on your Amiga back in the early 90s, chances are that PeaZip can still open them.

PeaZip can use GTK or Qt for its interface, and along with Deb and RPM packages, there are “portable” versions that can be run without installation. Just extract the **.tar.gz** file, jump into the resulting directory, and run **peazip** inside. Packaging up software to run across the myriad Linux distros out there is

tiring work, so kudos to the devs for putting a lot of effort in.

PeaZip’s interface is attractive, clean and largely self-explanatory, so we won’t dwell on it here, but instead focus on the new features. The biggest addition in 5.3 is the Compression Profiles. Go to File > Create Archive, and in the dialog that appears, click on the down arrow to the left of the OK button. This provides some pre-created settings for compression, such as “7zip with password” or “Keep output under 10MB for email attachments”. These shortcuts are very handy when you don’t have time to poke around inside individual settings boxes.

“PeaZip’s interface is attractive, clean and self-explanatory.”



Compression Profiles make it a one-click job to choose formats and options.

Also in 5.3 are improvements to the file manager (such as a better treeview sidebar), while bookmarks can be sorted. The inclusion filters that you can use when choosing which files should go in an archive are more flexible too. PeaZip just keeps getting better and better – it’s an outstanding piece of work.

PROJECT WEBSITE
<http://peazip.sourceforge.net>

Password manager

gpgpwd 0.4

In the wake of the Heartbleed OpenSSL vulnerability most major sites and services have recommended that users change their passwords – and, of course, it’s a bad idea to use the same password across multiple sites. So we’ve all ended up with another bunch of passwords to remember, and for those of us without incredible memories, a password manager comes in mightily useful.

gpgpwd is a great little manager written in Perl and requiring the JSON and Try-Tiny modules, which are available as **libjson-perl** and **libtry-tiny-perl** in Debian/Ubuntu-based distros. Extract the program, switch into the resulting directory and install as follows:

```
tar xfv gpgpwd-0.4.tar.bz2
cd gpgpwd-0.4
sudo make install
```

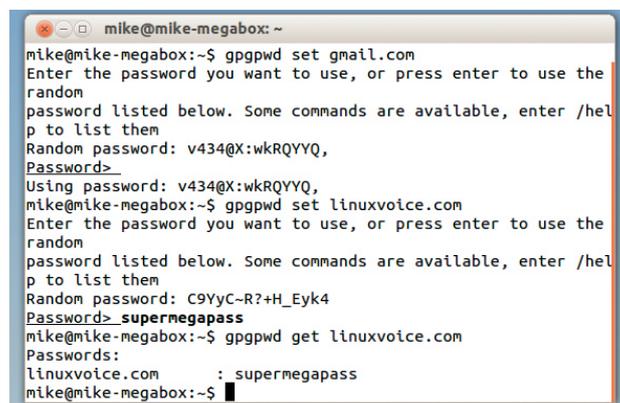
If you’ve used the GPG encryption tools before, you’re ready to go – but if not, enter **gpg --gen-key** to create a new encryption key. You’ll be prompted for a password, and this will become the master password you’ll use to access other passwords via gpgpwd. To create a new entry in gpgpwd’s database:

```
gpgpwd set somesite.com
```

Here, gpgpwd will ask you for a password for **somesite.com**, or offer to create a new one. A few commands are available for generating passwords: **/alphanumeric 20**, for instance, will generate a random 20-letter/number password. Hit Enter, and the password will be stored. To retrieve it in future:

```
gpgpwd get somesite.com
```

When you do this, you’ll be prompted for your master



gpgpwd is low on dependencies, uses existing tools (GPG) for security, and works without any faffing around.

password – that is, the GPG password you used during the **--gen-key** command earlier. Passwords are stored in the encrypted **.gpgpwddb** database file in your home directory, and to remove them from the database, use **gpgpwd remove** followed by the site name.

PROJECT WEBSITE
<http://random.zerodogg.org/gpgpwd>

File integrity checker

Checkit 0.2.0

If you've ever downloaded a large file such as a distro CD/DVD ISO image, you'll probably have seen a file called MD5SUM (or SHA256SUM) to go alongside it. This contains a checksum – a sequence of characters that can be used to check the integrity of the download. This is all good, but if you need to watch the integrity of many files on your filesystem, it becomes a bit fiddly to have **foo.md5sum** and **bar.md5sum** files all over the place. And then you have to run the md5sum tool each time... Checkit makes the whole process a lot easier by embedding a checksum into a file's extended attributes (so you don't need separate files) and letting you do batch checks on multiple files.

Its only dependency is **libattr1-dev**, and you install it with the usual **./configure, make** and **make install**

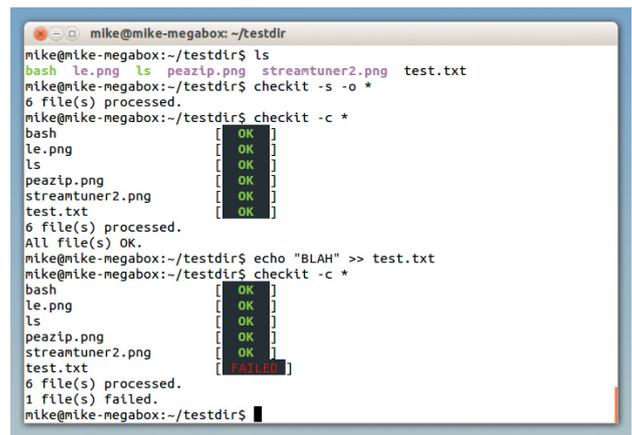
(as root) procedure. Go into a directory containing files that you want to check, and run this:

checkit -s -o *

This stores and overwrites the checksum for all the files in the current directory, storing the checksums in the file's extended attributes (ie the bit of the filesystem that contains metadata about a file, such as its creation date). Your filesystem needs to have extended attribute support for this to work, but if you're using ext3/4, XFS, JFS or BTRFS you'll be fine. Now modify one of the files, and then run:

checkit -c *

This checks all files against their



After we've modified test.txt, Checkit gives us a red "failed" message to show that the file has changed.

checksums, and shows you which files have changed.

Checkit isn't a replacement for a full-on intrusion detection system, and it's limited to certain formats and filesystems (many archive formats don't save extended attributes, for instance). But for simple jobs it works just fine.

"Checkit embeds a checksum into a file's extended attributes."

PROJECT WEBSITE
<http://dennisk.customer.netSPACE.net.au/checkit.html>

Text editor

LE 1.15

In last issue's cover feature we gave mad props to Vim, but we appreciate that not everyone likes its mode-based operation and (initially) confusing keybindings. Emacs is a hugely powerful alternative, but it can still be tricky to learn all the Ctrl/Meta+key commands. If you've come from an MS-DOS background, you might want a more accessible editor that presents its options in familiar menus – and LE could be exactly what you need.

Despite having decent knowledge of C, we couldn't get past the fiddly errors that cropped up when we tried to compile LE from source, so we converted the RPM package from <http://lav.yar.ru/download/le/binaries/> to a Deb for our Ubuntu system using **alien**. This is a very handy tool for converting packages

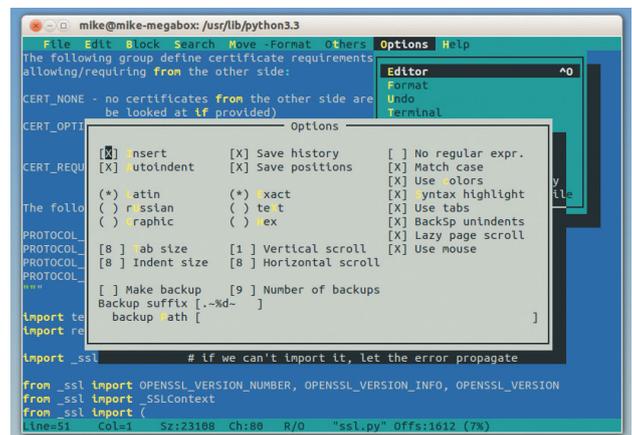
between formats. We used:

alien --to-deb le-1.15.0-1.x86_64.rpm

Along the bottom you'll see a status line containing information about the current file, while hitting F10 (or Ctrl+B) opens the menu at the top. Use the arrow keys to navigate through the menus, and enter to open them. You'll also see keyboard shortcuts in the menu – eg F2 to save, or Ctrl+F to search.

Back to the old house

LE includes all the common features you'd expect in a decent editor: syntax highlighting, search and replace with regular expressions, line and column selections (with many available operations to perform on them), UTF-* support, custom colours and more. It can be used as a hex editor (see the **-h** option), and even lets



LE is reasonably configurable, and if you don't like the default colour scheme, others are available.

you convert between Unix and DOS-style text files (they have different line ending characters).

While it's not the ultimate kitchen sink editor like Emacs, it does 95% of what most people need, and the menu-based interface makes it easy to pick up.

PROJECT WEBSITE
<https://github.com/lavv17/le>

FOSSPICKS Brain Relaxers

Board game compilation

Holtz 1.4.0

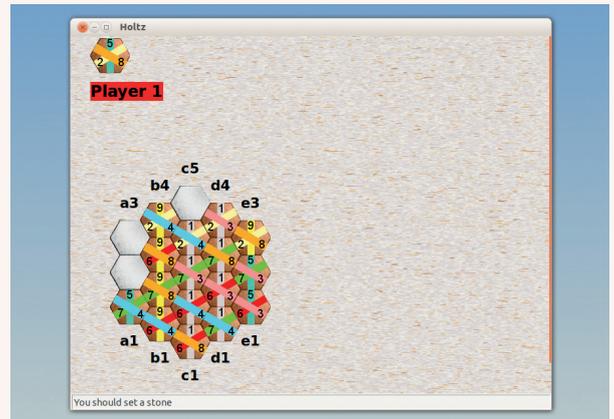
Holtz isn't a single game as such – it's an engine for four board games. And these aren't your common-or-garden board games either, but abstract strategy games that take a while to get your head around. Sadly, the README file is blank and the INSTALL file contains generic GNU build instructions from 2002, so here's how to compile it: install the wxWidgets development packages from your distro's package manager (eg **libwxgtk2.8-dev** and **wx2.8-headers**) along with Boost (**libboost-dev**). Then run **./configure, make** and **sudo make install**, followed by **holtz** to start the game.

Click **File > New** in the menu to choose one of the four included

games. The first, Zertz, is designed for two players, but if you're on your own, you can set up the computer as an AI player. It combines elements of draughts and solitaire, and every time you lay a piece, you remove a space on the board, so the playing area gradually gets smaller. (The rules are complicated, so click Help > Contents in the menu to get the full description.)

Game of four halves

Dvonn, meanwhile, is another two-player game, which involves stacking pieces on top of one another to control areas of the board, while Relax is a good single-player game where you have to score points by placing colour-aligned pieces. The final game, Bloks, is for two or four players and



This is Relax, where your goal is to place pieces so that the coloured lines extend for as long as possible.

involves racing from the corners to grab as much space on the board as possible.

Holtz is a brilliant little package, and some of the games will keep you busy for hours, even if they seem rather odd at first.

PROJECT WEBSITE
<http://holtz.sourceforge.net>

Tetris variant bundle

Galois 0.3

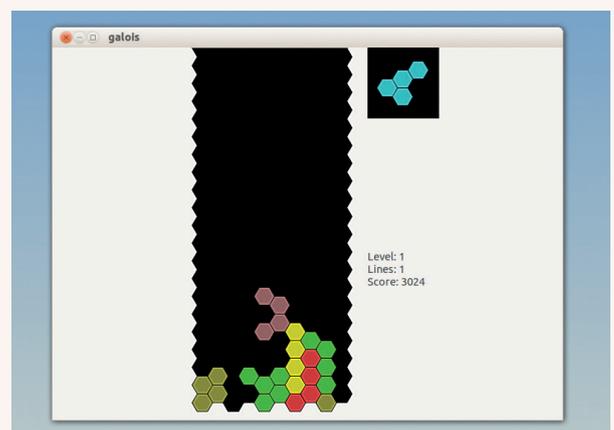
Tetris has pretty much been done to death at this point, although we still see new versions on **Freecode.com** every week. So at first we almost skipped over Galois, but then we saw the screenshots, and just had to try it. Galois is a Tetris engine that includes the classic game, but also some variants with different shapes. To compile it, you'll need GTK 2.4 and LibXML++ 2.6; in Debian and Ubuntu-based distros, these are provided in the **libgtkmm-2.4-dev** and **libxml++2.6-dev** packages.

When you start Galois, you're presented with a grand total of nothing. Just a grey screen. Click **Game > New** in the menu, however, and a version of the

regular Tetris game will kick off. By default, Galois's Tetrisy games run at a fair pace, but you can change that by going to Game > Stop and then Game > Preferences (see "Initial speed level"). This is also where you'll choose a different format, such as hexagonal or triangular bricks.

I'll have a T please, Bob

We love the hexagonal version: it's similar enough to Tetris that it doesn't take long to pick up, and many of the tactics you use in the classic game still apply. The controls are as usual for Tetris, but the unusual shapes make the game different enough to exercise other neurons in your noggin, and especially difficult at times too.



What's this – Tetris? Or Blockbusters? Or some ker-azy mutant hybrid of both?

There's only one flaw: pressing Esc to pause leaves the screen intact, so you can plan your next move (the screen goes blank in most Tetris variants). We'll submit a bug report... 🐛

PROJECT WEBSITE
www.nongnu.org/galois



LIBREOFFICE 4.2

FAST, COMPATIBLE, INTEGRATED

LibreOffice 4.2 offers performance and interoperability improvements, that are particularly appealing for power and enterprise users. **You can read and write old and new Microsoft Office files, and import Microsoft Publisher, Microsoft Visio, Corel Draw and Apple Keynote documents.** And you can do it faster than ever!



Test our Impress Remote Control for Android and iPhone/iPad available on Google Play Store and iTunes Store



Support The Document Foundation with a donation at <http://donate.libreoffice.org>



Download LibreOffice 4.2:
<http://www.libreoffice.org/download/>



LibreOffice
The Document Foundation

AND THANK YOU FOR USING LIBREOFFICE!

www.libreoffice.org



LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM



TUTORIALS

Dip your toe into a pool full of Linux knowledge with nine tutorials lovingly crafted to expand your Linux consciousness.



Ben Everard

Is off to investigate Linux usage on the Mull of Kintyre. Next issue: the frog chorus.

Money has always been a thorny subject in the Free Software world. There's no doubt that some companies are making huge amounts of profit on the back of open source software. In fact, we can't think of any major technology companies that don't. However, many of the projects that support these companies are chronically short of money. OpenSSL is a famous example of this, but there are many more.

Fortunately, it seems to be sinking in that essential infrastructure needs to be funded properly. Pitivi, The Blender Foundation and MediaGoblin have all raised significant amounts of money in crowdfunding. Elementary OS is working with BountySource.com to provide cash incentives to people fixing bugs. Hopefully, 2014 will be a turning point in the funding of Free Software.

Much as we may like the thought of developers working on open source projects for nothing, the truth is that complex software requires time and even the most altruistic people have bills to pay. In the past, most paid open source developers were paid by corporations and followed their agenda. By funding development directly, you get a say in what direction the project takes, and you can help struggling projects keep developing great software.

ben@linuxvoice.com

In this issue...



Create a font

Fonts are the handwriting of computers. Create your very own personal typography, with **Mike Saunders** and BirdFont.



Build a robot

Transform an ice cream tub, a Raspberry Pi, a mobile phone and two motors into a robot in just six pages full of **Ben Everard's** words.



Stalk wildlife

If you want to take pictures of badgers but you're afraid of catching bovine TB, follow **Jon Archer's** example and build a remote camera.



Harden a server

Linux can be secure, but it isn't always. Here's what you can do to prevent nasty hackers and botnets from getting into your servers – by **Mike Saunders**.



Move from XP

It's the OS that we love to hate, and it's embedded itself so deeply that it's hard to get rid of. **Mark Crutch** keeps XP alive (undead?) in VirtualBox.



von Neumann

Juliette Kemp moves on to John von Neumann, the architect of the modern computer, inventor of cellular automata and developer of the first virus.

PROGRAMMING

FPGAs

98 Design your own microchips. Really! With an FPGA board you can load new designs onto Field Programmable Gate Arrays and implement your own integrated circuits. We start simple and investigate some chips using the ZPU processor, then build a few circuits that use them.

Error detection

102 Data errors are an unfortunate fact of life. However, that doesn't mean we should just sit back and meekly accept our fate. With a bit of clever maths, we can fight back against flipped bits and ensure that the data that arrives is the same data that we send, because no-one likes nasty surprises.

Android Studio

104 Google has just released a new development environment: Android Studio. It's based on IntelliJ IDEA, and it looks great. We take a look around the preview version and show you how to create a simple smartphone app to help you digest all the latest news from the Linux Voice RSS feed.

MAKE YOUR OWN FONTS WITH BIRDFONT

MIKE SAUNDERS

You don't need to be a design whizz to create your own custom fonts – BirdFont makes it easy as a particularly good-looking pie.

WHY DO THIS?

- Create funky typefaces from scratch or based on existing designs
- Give your printed documents or website a unique feel
- Export to TTF, EOT and SVG formats

If you were using Linux in the late 90s (or you've seen screenshots of the desktop environments back then), you'll know that it was pretty ugly. Fonts, in particular, were a bit of a disaster area. Today we have gorgeous desktops and window managers, and distros ship with oodles of top-quality, free-as-in-freedom fonts. But have you ever considered making your own font? You can create one from scratch if you're full of ideas, or base one on an existing design – eg an old document or a logo. It's much simpler than it sounds, so we'll explain how.

To make our custom font we'll be using BirdFont (www.birdfont.org), an excellent font editor that runs on Linux, Mac OS X and Windows. Packages are available for many distros, but if you can't find it in your distro's repositories, grab **birdfont-0.37.tar.gz**



The street sign we'll be using to create the lowercase "a" character in our custom font.

from the project's website, extract it, and follow the instructions in the README. Once you have it installed, just enter **birdfont** in a terminal to start it.

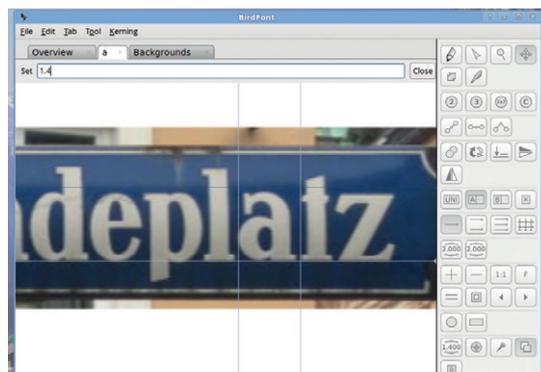
In this tutorial we'll use an existing design as the basis for a font. We'll take a street sign and create a glyph (font character) of the letter "a" from it. Of course, if you want to make a complete font then you'll need an image that contains all letters (uppercase and lowercase) along with numbers.

Step by step: create a font

1 Align image

Start BirdFont and click on File > New to create a new font. A list of glyphs will appear – scroll down and double-click on "a". In the right-hand toolbox, click the  button (it shows an uppercase B) towards the bottom to insert a new background image (all of the buttons have tooltips, so hover over them with the mouse to find out what they do).

Click on the + button to add an image, and then double-click its thumbnail. Move the image using the target () tool until the image's "a" character is inside or over the box. Right-click the  button to open a scale value bar, and scale the "a" until its height matches the box. Finally, grab the right-hand guide line using its small arrow at the bottom to match the "a" character's width.



We've moved and resized the image so that the "a" is inside the box, and pulled the right-hand guide in.

2 Create the outline

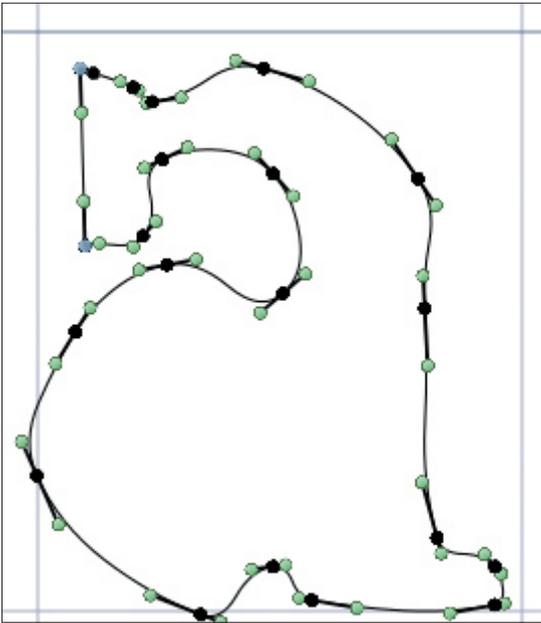
Now click the  icon (add new points) in the top-left of the toolbox, and click several times around the outside edge of the character to create an outline, eventually clicking on the first point to complete it. This outline can be pretty rough – you don't need to add points for every tiny detail. Use Shift+Ctrl+= (equals key) to zoom in.



Figure 2: The outline for our glyph. It's looking rather angular at this stage, but we'll fix that in a moment...

3 Smoothen the edges

Back in the bottom-right of the toolbox, click the  icon (show/hide background image). Then click the  button at the top. Now hold down Shift, and click on all of the blue points on the outline, going round the whole glyph clockwise. When they're all selected, click the  (tie curve handles) button in the tool pane and the edges will be rounded out.



That looks a lot better! With the edges rounded it's starting to look like a proper character.

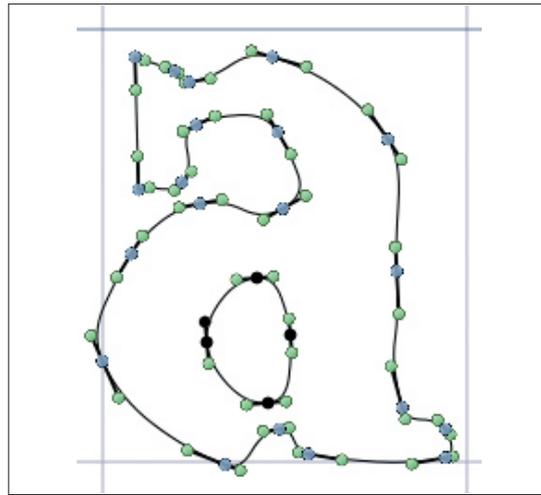
5 Preview it

When you're happy with everything, go to File > Preview (you'll be asked to enter a name for the font). Then a Preview tab will appear, showing your glyph being used in some example sentences.

If you're happy with the results, congratulations – you can now go on to do all the other letters! (It might be a long job.) If you need to fine-tune the character more, click its tab again, choose the arrow (Move Points) button in the tool pane, and fine-tune it. And if you need any help, pop by our wonderful forums at <http://forums.linuxvoice.com>. 

4 Align the paths

Click the  (show/hide background) button again. Chances are that the current paths won't be 100% perfectly in sync with the original image, so click and drag the blue points to line them up (they're Bézier curves, so you can also alter them with the green points). If you have an area that needs removing, like the hole in the bottom of the "a" character, for instance, draw a new path as per the previous instructions, and when it's complete click on Create Counter From Outline). Then smooth out the points as in step 3, to get the result shown below.



Here we've added the inside part of the character as a counter path. We won't be giving up the day job.



Our new "a" character in BirdFont's Preview. Sure, it looks rather out of place, but when we've done the others...

Exporting your design

When you've finished designing your font, click File > Export in the menu and provide a name for it. BirdFont will save your work as **<name>.bf** in your home directory (eg **/home/mike/myfont.bf**). It will also create various font files that you can install into your Linux distribution (or indeed other operating systems): **Typeface.ttf** (TrueType, the most common format), **Typeface.eot** and **Typeface.svg**.

It's also possible to include your font in your website, giving it a more personal feel than those sites that use regular Helvetica or Times fonts. During the Export process, BirdFont also generates a **Typeface.html** file. Have a look inside it, especially the `@font-face` parts of the CSS towards the top, to see how to use custom fonts in a page.

```
<style type="text/css">
  body {
    text-rendering: optimizelegibility;
    font-feature-settings: "kern";
    -moz-font-feature-settings: "kern=1";
    -ms-font-feature-settings: "kern";
    -webkit-font-feature-settings: "kern";
    o-font-feature-settings: "kern";
  }
  @font-face {
    font-family: 'TypefaceSVG';
    src: url('Typeface.svg#Typeface') format('svg');
  }
  @font-face {
    font-family: 'TypefaceTTF';
    src: url('Typeface.ttf') format('truetype');
  }
  @font-face {
    font-family: 'TypefaceEOT';
    src: url('Typeface.eot');
  }
</style>
```

BEN EVERARD

RASPBERRY PI: BUILD A MARS ROVER

Polish your CV and call NASA: you're about to become a professional-grade robot builder.

WHY DO THIS?

- Get started with robotics
- Learn more about the Raspberry Pi
- Build a robot army and take over the world

“**R**obotics is a complex area that requires a combination of electronics understanding and the ability to use specialised machinery”. That last sentence is a common sentiment, but it's utter balderdash. Modern development boards like the Raspberry Pi (and the host of expansions that do with them) combined with the flexibility of Linux makes robotics incredibly easy.

To prove this, we're going to build a Mars rover-type buggy based on a Raspberry Pi. You'll be able to control it remotely, and it'll stream video back to the controller. To make control really easy, we'll build a smartphone app to use the phone's accelerometer, so you can drive the buggy by turning the phone (much like the controls in many smartphone video games).

There are quite a few parts to this, and we'll be using a few different technologies to control different parts, but thanks to the wide range of development tools on Linux, it's not as difficult as it sounds. For the hardware you'll need:

- Raspberry Pi and SD card (it is possible with a model A, but a model B will be easier to develop on).
- Raspberry Pi camera module (the NoIR module will be able to see in the dark).
- Raspberry Pi-compatible Wi-Fi dongle (see http://elinux.org/RPi_USB_Wi-Fi_Adapters).
- Power supply for Raspberry Pi.
- Power supply for motors.
- Two motors and drive train.
- One or two more wheels.
- Motor controller.
- Chassis.

You'll also need a Linux machine to do some development on, and an Android phone (other smart

phones should work, though you'll need the appropriate development environment).

If you haven't worked with robotics before, the final four might sound a little complex, but don't worry, they needn't be. While you could use almost any motors you can get your hands on, there are some easy, reasonably priced ones that are particularly easy to use from PiBorg (<http://piborg.org/accessories/dc-motor-gearbox-wheel>) and other suppliers. You only need two of these to drive the robot, and the only assembly is pushing the wheel onto the axle.

Reliant Mars Robin

For the final wheel (ours has three, but yours could have four), we used a ball caster (like this one: <http://shop.pimoroni.com/products/pololu-ball-caster-with-3-4-metal-ball>). This allowed the back of the buggy to move freely and follow the front two wheels.

The Raspberry Pi does have General Purpose Input and Output (GPIO) pins that can be used to switch low-power components like LEDs on and off. However, motors draw a much higher current than the GPIO pins can provide. Therefore you need some way of taking a signal from the Pi and converting it into an electrical current powerful enough to drive a motor. For the purposes of this project, we can classify these into two types: on/off controllers and variable speed controllers. The first (such as the PicoBorg or the relays on a PiFace) will work, but the controls won't be as finely-grained as they could be. We used a PicoBorg Reverse (<http://piborg.org/picoborgrev>), which enables us to vary the speed of each motor (other controllers are available with the same features). The most important thing is that the board you use as the brains of the robot should be controllable from Python (almost all are). There should be sample code on the board's website to show you how to do this.

The build

The chassis can be as simple or as complex as you like. Specialised robot chassis are available that are robust and capable of carrying lots of sensors. We don't need this much for a simple buggy though. You can use anything provided you can mount the wheels on it and it will support the electronics.

Finally, we used a USB power pack and a 9V battery to power the Pi and the motors respectively. This is quite a lot of hardware, but all of it could be used on other projects.



An ice cream tub makes a simple and cheap robot chassis – just make sure you wash it out first.

Obviously the build will vary depending on exactly what parts you've chosen. For us, it involved connecting the PicoBorg Reverse according to the instructions on the website (<http://piborg.org/picoborgrev/install>).

To set up the buggy, we glued the motors to either side of one end of the ice cream tub, and bolted the caster to the other end. This created a three-wheeled buggy driven by the two motors at the front. We set the Wi-Fi to automatically connect to our network using the WiFi Config tool on the Raspbian desktop.

All motor controllers should come with some test code so that you can make sure everything is working. The software that installs the PicoBorg Reverse drivers will also put an app on the desktop. If you haven't already, you should run that now. Now is also a good time to make sure that both motors are wired the correct way round. With both motors on forward, the buggy should obviously move forward. With motor 1 on and motor 2 off it should turn left, and with motor 2 on and motor 1 off, it should turn right. If this is different on your buggy, you just need to switch the wires around until it works correctly.

Fire up Python

The PicoBorg Reverse software includes a Python module to control the motors, but it doesn't install it to the global Python directory, so it's not available to scripts that are run from other locations. In order to make this module available, you'll have to copy it across yourself with the following code (you may need to adjust the path depending on where you unzipped the install files):

```
sudo cp /home/pi/picoborgrev/PicoBorgRev.py /usr/lib/cd
pymodules/python2.7/
```

We'll use a simple web server to control the buggy. Web servers work by waiting for requests, and then serving web pages based on the request they get. Normally, the request is given in the URL that the website visitor's browser sends to the web server. For instance, if you visit www.linuxvoice.com/wp-

Alternatives to the Pi

The Raspberry Pi is particularly well suited to this project because the camera is well supported and there are plenty of motor control add-ons to provide all the functionality you need. However, it's not the only option. It should be possible to do more or less the same thing on a BeagleBone Black, although you'll have to do a little bit more work to get streaming video set up (there's a guide here: <http://shrkey.com/installing-mjpg-streamer-on-beaglebone-black>). Larger boards such as the Odroid or Udo0 should work as well, though they'll drain the batteries faster, and their extra processing power isn't really useful for this project.

It should be possible to use a microcontroller such as an Arduino to handle the motor control (though it would be better to use Bluetooth than Wi-Fi in this case). Getting streaming video working with a microcontroller would be challenging, though probably not impossible if you are determined enough. However, you could do this separately using a wireless webcam.



A bit of glue will hold the motors in place, but be careful not to get any on moving parts.

`content/uploads/2014/04/turtle.png` you are requesting the file `/wp-content/uploads/2014/04/turtle.png` from the server www.linuxvoice.com. The server will respond to this request by sending an image from the Python drawing tutorial from Linux Voice issue 2.

Requests don't have to be for files though. The web server can deal with requests however it wants. You can also send bits of data in the URL. These arguments in the URL string come after a question mark and are separated by ampersands. For example, in the URL www.google.co.uk/search?q=linuxvoice, the argument `q` is set to the string "linuxvoice".

We're going to use the Python Tornado web server to use these requests to control the motors on the Pi. You'll need to install this on the Pi with:

```
sudo apt-get install python-tornado
```

The code to control the motors using the PicoBorg Reverse is:

```
import PicoBorgRev
import subprocess
import tornado.ioloop
import tornado.web

maxspeed = 0.3
PBR = PicoBorgRev.PicoBorgRev()
PBR.Init()
PBR.ResetEpo()

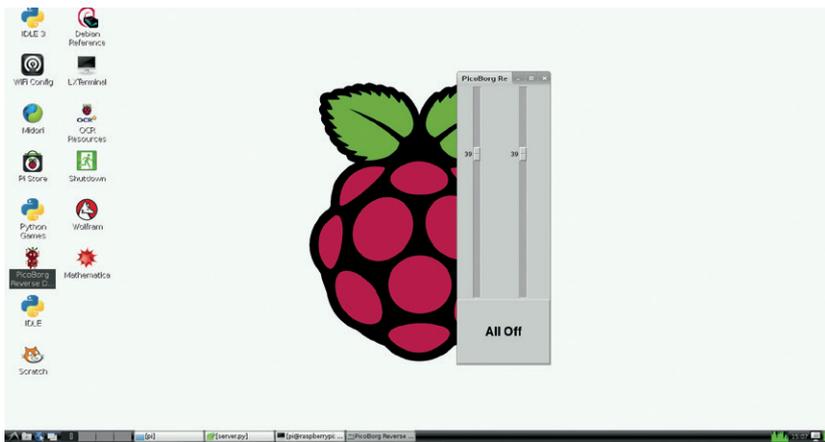
class TurnHandler(tornado.web.RequestHandler):
    def get(self):
        PBR.SetMotor1(min([float(self.
get_argument("motor1"))/100, maxspeed]))
        PBR.SetMotor2(min([float(self.
get_argument("motor2"))/100, maxspeed]))
        self.write("Updated")

class HaltHandler(tornado.web.RequestHandler):
    def get(self):
        subprocess.call(["sudo", "halt"])

if __name__ == "__main__":
```

LV PRO TIP

Robots are like Lego: once it's built, play with it for while, then take it to bits and build a new one.



The PiBorg Reverse GUI controller is useful for making sure everything's connected correctly.

```

application = tornado.web.Application([
    (r"/turn/", TurnHandler),
    (r"/halt/", HaltHandler)])
application.listen(8000)
tornado.ioloop.IOLoop.instance().start()
    
```

The final block of this code (which starts with `if __name__`) sets up the web server running on port 8000 (we'll use port 80 – the usual web server port – a bit later). It uses the class `TurnHandler` to handle requests to `/turn/`, and the class `HaltHandler` to deal with calls to `/halt/`. Both of these classes extend `tornado.web.RequestHandler`, which sets them up with almost everything they need. The only thing this code does is add the `get` method that is called whenever a HTTP GET request is sent to the appropriate URL.

You can access the arguments passed in the URL using the `self.get_argument()` method. The two calls in `TurnHandler` are to get the arguments called `motor1` and `motor2`. We then use these values (which

we'll set between -100 and 100) to set the speed of the motor (which is between -1 and 1). We've limited the motor speed using

the global variable `maxspeed` to stop the motors burning out.

The code here works for a PicoBorg Reverse, but it should be fairly trivial to adapt it to other motor boards. If your motor controller only supports on and off, you'll have to include an `if` statement to test the

“You could add an output device to the Pi such as a little LCD screen to display the IP address.”

arguments against a threshold, and if it is, turn the motor on. For example:

```
if float(self.get_argument("motor1")) > 30.0:
```

```
    #code to turn motor one on
```

```
else:
```

```
    #code to turn motor one off
```

`HaltHandler` is used to turn the Pi off, since there's no other way to shut it down cleanly when there's not a screen unless you SSH in, which is a little excessive for a simple robot.

We called the file `server.py`, and you can start it running from the LXTerminal command line with:

```
python server.py
```

We'll get it running automatically a bit later on.

You can now control the robot from the Raspberry Pi by opening the web browser and going to `http://localhost:8000/turn/?motor1=20&motor2=20` (be careful not to accidentally drive your robot off your desk when testing this). You can then stop the motors by going to `http://localhost:8000/turn/?motor1=0&motor2=0`.

Control from other machines

You can also access this from other computers on the same network by using the IP address of the Pi. To find out the IP address of the Pi, open LXTerminal and type `ifconfig`. This will output a block of information for each of the network interfaces. The one you need is labelled `wlan0`, and you're looking for the `inet addr`. In the following, the IP address is 192.168.0.33:

```

wlan0  Link encap:Ethernet HWaddr bc:ee:7b:87:7b:38
        inet addr:192.168.0.33 Bcast:192.168.0.255
        Mask:255.255.255.0
        inet6 addr: fe80::beee:7bff:fe87:7b38/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500
        Metric:1
        RX packets:88425 errors:0 dropped:0 overruns:0
        frame:0
        TX packets:81516 errors:0 dropped:0 overruns:0
        carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:76786575 (76.7 MB) TX bytes:14405224
        (14.4 MB)
    
```

Unfortunately, this isn't fixed and may change from time to time if you reboot the Pi, and it won't be easy to run `ifconfig` if the Pi is mounted inside a robot.

There are a few ways around this. Many Wi-Fi routers enable you to assign a static IP address to a device,

Web sockets

The method we've used for controlling the motors is, well, a little hacky. It works, but it doesn't work well. The main problem is that there's a large overhead each time you change the motor speed. The phone app has to negotiate a new TCP connection and send the data, then the Tornado server re-initialises the module to send data to the server. This means there's a noticeable lag between turning the controls and the buggy responding. Part of this is also due to the interval that the app checks the accelerometer, but this has been adjusted to work well with the speed of the server.

A better method would be to create a communications channel through which you can continuously send data. There are a couple of options for this: TCP sockets or Web sockets. Both are supported by Python, and both have plugins for the Cordova framework that we're using for the Android app. Neither should be excessively complex to set up, though they will require some knowledge of both Python and JavaScript. Using one of these methods, you should be able to reduce the latency of the control and increase the frequency with that the app updates the accelerometer readings.

which will enable you to set it so the same IP address will always be assigned to the Pi. You could add some output device such as a little LCD screen to the Pi to display the IP address. The simplest method is to use another Linux computer to scan the address range and find the IP address for the Pi. You can do this using Nmap.

First you'll need to install Nmap from your distro's repositories (on Debian-based systems, this is done with `sudo apt-get install nmap`). Since the above server runs on port 8000, we can use this to detect the Pi. The following command will check all computers in the IP range 192.168.0.0 to 192.168.0.20 to see if that port is open.

```
nmap -sT 192.168.0.0-20 -p 8000
```

The Pi will respond with something like this:

```
Nmap scan report for 192.168.0.33
```

```
Host is up (0.039s latency).
```

```
PORT      STATE SERVICE
```

```
8000/tcp  open  http-alt
```

Usually, the Pi will be the only IP address that returns a state of **OPEN** for this port.

Currently, you also need to start `server.py` manually. We'll set it to start automatically at the end once everything else is set up.

Getting visuals

Installing the Raspberry Pi camera module is simply a case of slotting it into the correct port (the one between the Ethernet and HDMI ports) with the silver coloured bare metal facing towards the HDMI port, then enabling it. Enter `sudo raspi-config` in LXTerminal, then select Enable Camera, then Yes. You'll need to reboot the Pi for the changes to take effect. There's a video guide at www.raspberrypi.org/help/camera-module-setup if you have any problems.

If you don't have a camera mount to attach to the chassis, a blob of Blu-tack also works.

That's the hardware completely set up. There's still a little bit of software to set up on the Pi, but it doesn't involve any more coding. As the saying goes, "good programmers borrow, great programmers steal", and that's exactly what we're going to do. Streaming video from a Raspberry Pi to a website isn't new, and there's no reason to do it yourself.

The easiest setup we've found is at https://github.com/silvanmelchior/RPi_Cam_Web_Interface. Just download the ZIP file and install it with:

```
unzip Rpi_Cam_Web_Interface-master.zip
```

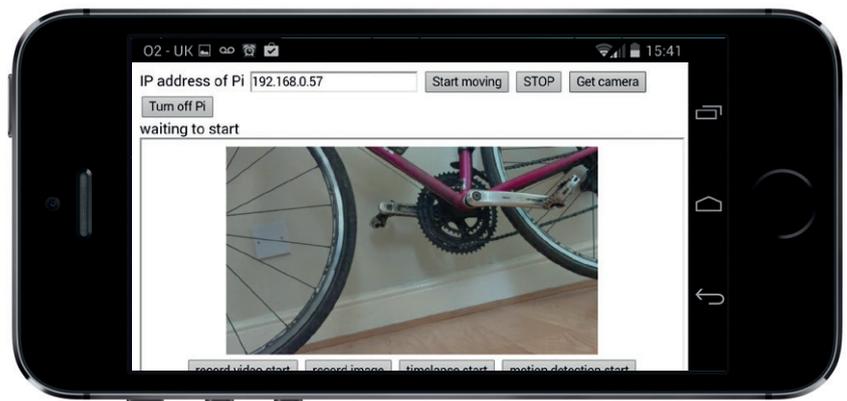
```
cd Rpi_Cam_Web_Interface-master
```

```
chmod a+x RPi_Cam_Web_Interface_Installer.sh
```

```
./RPi_Cam_Web_Interface_Installer.sh install
```

Reboot the Pi so it picks up all the new settings. It'll automatically create a web server (on port 80) that starts when you turn on the Pi, and hosts a website with the streaming video as well as some settings so you can control the video stream (and record pictures and video from your buggy).

Once it's up and running, you should be able to open `http://<ip-address-of-pi>` in a web browser on



The finished app controlling the buggy. It's not much to look at, but the controls are intuitive and fun.

another computer connected to the same network and see the video stream. You don't need to modify it at all, but it'll fit into the smart phone app we'll create in the next step a bit better if you get rid of the title and resize the image.

To do this, open up the `/var/www/index.html` file on the Pi using a text editor running as `sudo`. For example, to do this in Leafpad, run

```
sudo leafpad /var/www/index.html
```

To get rid of the title, delete the line:

```
<h1>RPi Cam Control</h1>
```

The size you want the image to be will depend on the resolution of your phone screen. We went with a width of 400 pixels, though you can adjust this at the end to make it fit properly on your phone. To do this, change the line:

```
<div><img id="mjpeg_dest"></div>
```

to:

```
<div><img id="mjpeg_dest" width=400px height=auto</div>
```

The only thing left to do set the Python script that runs the motor control server to start automatically (we didn't do this earlier because the setup for the webcam overwrites the file it's done in). Just add the following line (you may have to modify it depending on where you saved `server.py`):

```
python /home/pi/picoborgrev/server.py
```

to the file `/etc/rc.local` directly before the final line (exit 0). Again, you'll need to use a text editor running as superuser, so open Leafpad with `sudo` as you did with `index.html`. That's all the setup for the Pi – now to create the phone app that will control it.

Hands on

The easiest way to create smartphone apps is with Apache Cordova (as seen in Linux Voice issue 2). The idea is that it enables you to use web technologies (mainly HTML and JavaScript) to create apps that can access phone functions that regular web pages cannot. In this case, we'll access the accelerometer.

Accelerometers measure what's known as proper acceleration. This is a little different from what most people know of as acceleration, because it's the acceleration experienced by an object. This means that an accelerometer resting on a surface will experience an acceleration of 9.8 m/s because it's experiencing that acceleration from gravity. On the

other hand, if you drop the accelerometer, it will read 0 because it's in free fall and not experiencing any acceleration. (Actually, it will read a little higher than 0 because of air resistance.)

As long as you hold the device still, the accelerometer measures gravity. It measures it in three dimensions (x, y and z), which means that you can use it to measure the orientation of the device in three dimensions. In other words, it tells you which way up the device is.

“Cordova’s Accelerometer plugin should work on just about every smartphone.”

Accelerometer plugin should work on every phone that supports Cordova, which is just about every smartphone (Amazon Fire OS, Android, Blackberry 10, FirefoxOS, iOS, Ubuntu Touch, Windows phone 7 & 8, Windows 8 and Tizen). We'll look at Android here, and there are details of how to get started in the different environments on the Cordova website (http://cordova.apache.org/docs/en/3.4.0/guide_platforms_index.md.html#Platform%20Guides).

Cordova runs on node.js, so you'll need to install **npm** (the node package manager) from your distro's repositories. People using Ubuntu-based systems will need to add a PPA to get the most up-to-date version of node for this.

```
sudo add-apt-repository -y ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install npm openjdk-6-jdk
sudo npm install -g cordova
```

As well as Cordova, you'll also need the Android Software Development Kit (SDK) from Google (download this from <http://developer.android.com/sdk/index.html>). Once you've downloaded and installed this, you'll need to set up some environmental variables so that Cordova knows where to find

```
export PATH=${PATH}:/home/ben/adt-bundle-linux-x86-20140321/sdk/platform-tools:/home/ben/adt-bundle-linux-x86-20140321/sdk/tools
```

First, though, you'll need to set up a Cordova environment on your development machine. According to the Cordova documentation, the

```
export PATH=${PATH}:/home/ben/adt-bundle-linux-x86-20140321/sdk/platform-tools:/home/ben/adt-bundle-linux-x86-20140321/sdk/tools
```

You'll need to amend the paths to point to the Android SDK you downloaded and extracted. You can run these commands in the terminal, but it won't remember the settings, so you'll have to re-enter them each time you reboot. In order to add these permanently, add the two lines to the **.bashrc** file in your home directory.

To create a Cordova project for the buggy run:

```
cordova create buggy
cd buggy
cordova platform add android
cordova plugin add org.apache.cordova.device-motion
```

We based our code on the **watchAcceleration** Full Example from http://cordova.apache.org/docs/en/3.3.0/cordova_accelerometer_accelerometer.md.html#Accelerometer. This provides everything to read the acceleration periodically, and the function **onSuccess()** is called when it's successfully read.

Before getting into what we do with the acceleration, let's look at how we'll lay out the screen. This is the code between **<body>** and **</body>**:

```
IP address of Pi <input type="text" name="ip" id="ippi">
<button onclick="startMoving()">Start moving</button>
<button onclick="stopMoving()">STOP</button>
<button onclick="getCamera()">Get camera</button>
<div id="sendingstring">waiting to start</div>
<iframe id="cam" width=100% height=600px</iframe>
<iframe id="turniframe" width=1px height=1px</iframe>
```

As you can see, there will be a text field to enter the IP address of the Raspberry Pi, and three buttons to start controlling the motors, stop controlling the motors, and start the camera feed. **<div id="sendingString"></div>** will hold the URL that's being sent to control the motors. This isn't necessary, but it's useful to see what's going on.

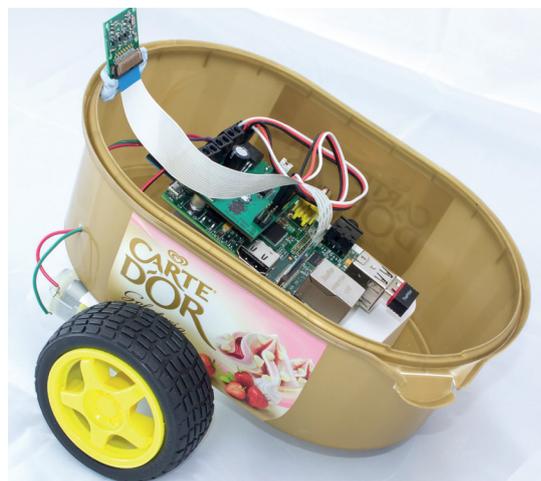
Embed video

Iframes enable you to embed web pages inside of web pages. The first one (with the id **'cam'**) holds the streaming video from the Raspberry Pi camera. The second one (with the id **'turniframe'**) doesn't actually hold anything useful, but by changing its URL, we can use it to create GET requests that control the motors.

To make this work, you need three new JavaScript functions that will run when the buttons are pressed:

```
function getCamera() {
    document.getElementById('cam').src = "http://" + document.getElementById('ippi').value;
}
function startMoving() {
    window.piMoving=true;
}
function stopMoving() {
    window.piMoving=false;
}
```

The first of these just sets the URL of the **cam** iframe to the address of the streaming webcam



That's all it takes to build a simple robot: Linux on the Raspberry Pi to power the motors, and Linux on a smart phone to handle the controls.

running on the Pi. Remember that we've removed the title and resized the image to make it fit in here. The rest of the controls are still there, so you can tune the streaming image by scrolling down the iframe.

startMoving() and stopMoving()

set the variable **window.piMoving** to **true** or **false**. This is just a global variable that we'll use to control whether the motor settings are sent to the Pi or not.

You also need to update the **onSuccess()** function (which runs every time it reads the acceleration) to:

```
function onSuccess(acceleration) {
    var element2 = document.getElementById('sendingstring');

    if (window.piMoving) {
        var motor1Prop = (acceleration.y + 10)/20;
        var motor2Prop = 1 - motor1Prop;
        var totalSpeed = acceleration.z * 10;
        var motor1Speed = motor1Prop * totalSpeed;
        var motor2Speed = motor2Prop * totalSpeed;
        sendString = "http://" + document.getElementById('ippi').value + ":8000/turn/?motor1=" + motor1Speed + "&motor2=" + motor2Speed;
        element2.innerHTML = sendString;
        document.getElementById('turniframe').src = sendString;
    }
}
```

Although it's not completely necessary, you can increase the frequency with which the app updates the buggy's speed by altering the frequency setting in the **startWatch** function. In the following example, it updates it once a second, but you could set this to be higher or lower.

```
function startWatch() {
    var options = { frequency: 1000 };
    watchID = navigator.accelerometer.
watchAcceleration(onSuccess, onError, options);
}
```

The full code is on the Linux Voice website.

This calculates the speed for the two motors. **acceleration.y** is used to change the direction and **acceleration.z** is used to change the speed. This works for holding the phone in landscape. With the screen at right angles to the ground, the buggy will stop, and as you tilt the screen forward (so the screen starts to face upwards), it will start to move. If you tilt the screen back, the buggy will move backwards. Tilting the screen from side to side (as though it were a car steering wheel) will turn the buggy.

Security

This robot is controlled via Wi-Fi with absolutely no security whatsoever. Anyone else on the network could quite easily take over control. Normally this isn't a problem on a local area network, but there may be occasions where you want a bit more privacy. Tornado does handle security quite well, though it's beyond the scope of this tutorial to go into it in detail. Take a look at the documentation on the project's website for guidance on this (www.tornadoweb.org/en/stable). Securing the video stream may be a little trickier, as it's not really designed for it.

```
index.html (-/buggy/www) - gedit
index.html (-/buggy/www) - gedit
index.html
36 window.piMoving=false;
37
38 }
39
40 function haltPi() {
41     document.getElementById('turnIframe').src = "http://" + document.getElementById('ippi').value + ":8000/halt/";
42 }
43
44 function stopWatch() {
45     if (watchID) {
46         navigator.accelerometer.clearWatch(watchID);
47         watchID = null;
48     }
49 }
50
51 function getCamera() {
52     document.getElementById('cam').src = "http://" + document.getElementById('ippi').value;
53 }
54
55 function onSuccess(acceleration) {
56     var element2 = document.getElementById('sendingstring');
57
58     if (window.piMoving) {
59
60         var motor1Prop = (acceleration.y + 10)/20;
61         var motor2Prop = 1 - motor1Prop;
62         var totalSpeed = acceleration.z * 10;
```

The acceleration in each axis is returned as a number between -10 and 10. The formula (acceleration.y+10)/20 returns a number between 0 and 1 depending on how far the phone is rotated. This is then used as a multiplier for the speed of one motor. The multiplier for the speed of the other motor is this value taken away from 1.

The overall speed is the acceleration in the z axis multiplied by 10. This gives it the range -100 to +100 (with negative values being backwards). This is the same range that the motors have. To get the final speed for each motor, we just multiply that motor's proportion by the total speed. This is quite a simplistic method of calculating the speed, and the turn directions will go back to front if the phone's held the wrong way up. However, it works, and it's easy to understand, so it's good enough for our buggy.

With the code ready, you just need to get it on to a phone in order to run it. Unfortunately, this can require a little fiddling with the Udev rules. There's full information on the Android developer site here: <http://developer.android.com/tools/device.html>. You can skip step 1 because Cordova will handle it for you.

Once this is set up, and the phone is plugged into your computer, you can compile and transfer it to the phone. Enter the following in a terminal in the root directory of the app:

```
cordova build android
```

```
cordova run android
```

As you can see, this isn't a particularly elegant solution. Running two web servers is a little over the top. It could have been re-written to do everything in one either by serving the video up from Python or by controlling the motors from PHP. The phone app could be more integrated rather than just serving up an iframe of the webcam controller. However, this project isn't about technical perfection, it's about demonstrating how you can quickly and easily link things together to easily create complex robots by using the tools that are available on Linux.

Ben Everard is the co-author of the best-selling book on learning Python with the Raspberry Pi, *Learning Python with Raspberry Pi*. He wrestles lions for fun.

We've used Cordova to create a phone app, but you could easily modify the code to create a web interface using sliders or buttons to control the buggy.

RASPBERRY PI: MONITOR WOODLAND CREATURES

JON ARCHER

Set up a sturdy camera out in the woods and use Linux to take pictures of lions, tigers and bears.

PARTS REQUIRED

- Raspberry Pi
- SD Card (bigger the better)
- Pi Camera
- Waterproof case with see through area
- USB Wi-Fi dongle
- USB rechargeable portable battery pack

A trail camera will capture images of wildlife that frequent a certain area, such as woodland. These images, still or moving, can be captured without the need of the photographer to be present. The camera either constantly records video or uses motion detection technologies to trigger an image capture. Off-the-shelf versions of these are expensive, don't offer any option for customisation, and contain proprietary hardware and software.

A Raspberry Pi, with its fantastic range of hardware and software options, is an ideal platform to create a similar device with the potential to create your very own wildlife videos and photographs. Although this project is geared towards building a Pi-based trail camera, there are many other situations where this could potentially be deployed; it would also make a simple security camera, for example.

Installation

First, ensure your camera is connected to the Pi and that you have a network connection for installation. Now we can install the OS and the required software.

The central part of this project is a piece of software called RaspiMJPEG (which is based on the MMAL library) to control the Pi camera.

From the starting point of a base Raspbian install, we can start up the Pi and use the configuration tool to increase the free space, ensure that SSH starts on boot and set the password for the Pi user. One other vital step is to enable the Pi camera.

Once we have finished with the configuration tool the next thing is to ensure that the system is up to



Components cost money, but the beauty of the Pi is that everything can be re-used for your next project.

date (**sudo apt-get update; sudo apt-get upgrade**). One more update we need to pull in is the latest firmware for the Raspberry Pi. This includes the latest camera firmware, which is required by the RaspiMJPEG camera control software. This can be done by running **sudo rpi-update**.

It should be part of the Raspian install, but we also need to ensure that Git is installed. This will be used to retrieve the software and scripts needed to complete this installation. Let's confirm it is installed by running the command **sudo apt-get install git**. We can now start to install the components required to provide the web interface, motion detection and live feed. Fortunately for us the majority of this is captured inside a script created by Silvan Melchior, who also created RaspiMJPEG. Run the command:

Git clone https://github.com/silvanmelchior/RPI_Cam_Web_Interface.git

This will download the initial scripts for the install along with some configuration files and pre-compiled binaries (don't worry – these are open source, just pre-compiled to save time).

Using a cat5 for power and connection.

If cat5 were an option, then powering the Pi could be achieved by using a power over Ethernet injection kit. Also, if the cable is run over a considerable distance then voltage drop must be taken into consideration. Therefore a higher voltage power supply should be used and a voltage regulator at the Pi side to ensure that it receives the necessary 5V.

The PVC box is one option for housing the project, but the recently crowdfunded kickstarter campaign PICE should also do the job quite nicely.

The box we used had an IP rating of 65, which essentially means it is nice and waterproof.



Software used in this tutorial:

- Apache httpd with PHP for the web interface
<http://httpd.apache.org> & www.php.net
- Motion, used for the motion detection
www.lavrsen.dk/foswiki/bin/view/Motion/WebHome
- Raspimjpeg – to interface with the camera and output as image/video/stream
www.raspberrypi.org/forums/viewtopic.php?t=61771

You should see an output similar to:

```
Initialized empty Git repository in /home/pi/RPi_Cam_Web_Interface/.git/
```

```
remote: Reusing existing pack: 161, done.
```

```
remote: Total 161 (delta 0), reused 0 (delta 0)
```

```
Receiving objects: 100% (161/161), 104.62 KiB, done.
```

```
Resolving deltas: 100% (70/70), done.
```

A new directory will be created called **Rpi_Cam_Web_Interface**. In this directory resides a script that will complete the rest of the installation. Use **cd** to move into the directory, and launch the script with **./RPi_Cam_Web_Interface_Installer.sh install**

This script will go away and install all the required packages, of which Apache HTTPD, PHP and Motion are the most important.

Afterwards we have a couple of choices as to how the software will start, if at all, on boot. For this tutorial we will have it automatically start with motion detection. For this we need to edit the file **/etc/raspimjpeg**, the last line of which contains the line **motion_detection false**

Use your favourite text editor to change this value to **true**, then re-run the install script with the option **autostart_yes** instead of **install** to set the software up to start on boot. That's all there is to the install. There is much that could be configured both within **/etc/raspimjpeg** or Motion, but for now we have a working system. Let's reboot!

Launch your favourite web browser from another PC and you should be presented with a live image from your camera with a series of buttons and a table of options underneath. If this is the case then our installation was a success.

Most of the buttons you see should be greyed out, with only the Motion Detection Stop button available. At this point you can test the motion detection by waving an object in front of the camera; subsequently clicking on the Download Videos And Images link you will see a video file listed.

Back on the RPi Cam control main page, the table of options presents a multitude of configuration; from here you can set image resolution, image quality, various levels of brightness, ISO, contrast etc. Experiment with these to find the best setup for you.

Enclosure and powering the device

As the Raspberry Pi and its camera will be outdoors, choosing a suitable enclosure is vital to ensure your Pi stays nice and dry. In our project we used a PVC outdoor electrical junction box (150x110x70 mm).



We were hoping to find badgers, but just got these deer. D'oh, a deer!

This box was all good and safe, but there was no opening for cables or view area for the camera to see out of. This is where a 45mm camera skylight lens and a hot glue gun came in handy.

If your Pi is already in a plastic case then simply glue this to the deeper side of the PVC junction box, otherwise some M3 size nylon stand-off spacers should be used to attach the board inside the box. You'll need to drill a hole into the junction box where the lens will be situated – ensure that when you attach the lens a liberal amount of glue and or sealant is used to ensure the box stays waterproof. Using one of the many available plastic camera mounts also helps with securing it inside the case with glue.

How you decide to power the device is all dependant upon the location you choose and the facilities available in that location. If your camera is to be situated in your garden or surrounding then laying a cat5 cable inconspicuously may be an option with some kind of power over Ethernet solution. Otherwise the Pi can be powered using a battery pack such as those used for emergency mobile phone charging, just make sure there is a reasonable capacity in the batteries such as 10,000mAh. If a battery pack is used then this must also be taken into consideration when deciding on an enclosure as extra room may be required. The downside to running on batteries would be that a live view would only be available if the box were be situated within the signal range of a wireless router. Using wireless would also have a bearing on the battery drainage and time available.

For simplicity we will power the Pi using a battery pack that will fit nicely inside the PVC box.

We won't go into the configuration of wireless dongles as this varies slightly for each device and is well documented, but once you have this configured and the battery pack fully charged, plug it into the Pi, secure your box and place it where you expect to see your target subject, then head back to your PC and watch the live feed through your browser. And don't forget to check the battery level regularly! 📱

Jon Archer is a Free Software evangelist, Red Hat ambassador and the founder member of RossLUG.

SSH, APACHE & TIGER: MAKE YOUR SERVERS SUPER SECURE

MIKE SAUNDERS

Lock down your Linux installations for maximum security and keep one step ahead of crackers.

WHY DO THIS?

- Stop bots and crackers getting easy access to your systems
- Understand the trade-offs between security and convenience
- Re-use the skills you learn here when you install distros in the future

Bruce Schneier, the well regarded American expert on cryptography and computer security, once said these wise words: “security is a process, not a product.” Keeping your servers safe from malicious types isn’t just achieved by chucking on a few extra pieces of software, but by having proper plans and procedures to deal with issues that come up. And security is a moving target – you might have your systems locked down and fully patched right now, but you never know what holes are going to be discovered in the future. Look at the OpenSSL Heartbleed mess, as an example...

Anyway, while most server-oriented Linux distros are pretty secure out of the box, they still make certain sacrifices for user-friendliness. In this tutorial we’ll show you how to tighten key components in a server system, including OpenSSH and Apache, and demonstrate how you can mitigate potential problems in the future with scanning tools and an intrusion detection system.

In this case we’ll be using a vanilla installation of Debian 7, as it’s arguably the most popular GNU/Linux distribution used on servers, but the guides here will be applicable to other distros as well.

1 HARDENING OPENSSSH

It’s absolutely imperative that we start with OpenSSH. Why that’s? Well, it’s almost certainly the way you’ll be interacting with your server, unless you have the luxury of logging into it directly via a physically connected keyboard and monitor. For headless servers, a good SSH setup is critical, because once you have that out of the way, you can focus on the other running programs.

OpenSSH’s daemon (server) configuration file is stored in **/etc/ssh/sshd_config**, so you’ll need to edit that (as root) to make changes to the setup. The first thing to do is find this line:

PermitRootLogin yes

Change **yes** to **no** here to disable direct root logins via SSH. This immediately adds an extra layer of security, as crackers will have to log in with a regular

```

sshd_config = (/etc/ssh) - VIM
File Edit Tabs Help
1 # Package generated configuration file
2 # See the sshd_config(5) manpage for details
3
4 # What ports, IPs and protocols we listen for
5 Port 22
6 # Use these options to restrict which interfaces/protocols sshd will bind to
7 #ListenAddress ::
8 #ListenAddress 0.0.0.0
9 Protocol 2
10 # HostKeys for protocol version 2
11 HostKey /etc/ssh/ssh_host_rsa_key
12 HostKey /etc/ssh/ssh_host_dsa_key
13 HostKey /etc/ssh/ssh_host_ecdsa_key
14 #Privilege Separation is turned on for security
15 UsePrivilegeSeparation yes
16
17 # Lifetime and size of ephemeral version 1 server key
18 KeyRegenerationInterval 3600
19 ServerKeyBits 768
20
21 # Logging
22 SyslogFacility AUTH
23 LogLevel INFO
24
25 # Authentication:
26 LoginGraceTime 120
/etc/ssh/sshd config [RO] 20,0-1 Top

```

A good Vim setup (see last month’s cover feature) provides syntax highlighting for **sshd_config**, making it easier to read and edit.

user account and password first, and then know the root password as well. (Warning: make sure you have a regular user account on the system first, because if you only have a root account, you can lock yourself out by changing this!)

Next, add a line like this to the configuration file:

AllowUsers mike graham ben

This restricts which users can log in via SSH; if you have many accounts on the machine but only one or two will log in, this is worth doing.

Next, change this line:

Port 22

22 is the standard SSH port, so it's a good idea to change this to something else (and make sure that your router or firewall is also aware of the change if you'll be logging in from outside your network). A random number like 1234 is fine here – it adds a bit of “security through obscurity”. When you log in with the **ssh** command now, you'll need to add **-p 1234** to the end of the command.

Triple lock

Now, these three changes are useful enough on their own, but together they add a major layer of protection against automated cracking scripts and bots. These are programs that attempt to break into your machine by repeatedly trying username and password combinations, many times a second, until they get access. (If you have a net-facing machine with OpenSSH that has been online for a while, look in **/var/log/auth.log** and you'll probably see many login attempts from IP addresses around the world.)

The default OpenSSH configuration means that these bots don't have to do much work: they know that the root account is available, and they know to try on port 22. By disabling root access and switching to a different port, the bots have to do a lot more guesswork, trying random ports and usernames. If you have a strong password, this makes it very difficult for a bot to gain access.

Once you've made your changes to **/etc/ssh/sshd_config**, you'll need to restart the OpenSSH daemon:

```
service ssh restart
```

Passwordless authentication

While good passwords are hard to crack, you can make it almost impossible for nasty types to log in by disabling password authentication, and using public/private key pairs instead. On the machine(s) you use to log in, enter **ssh-keygen** to generate the keys, then accept the defaults for the file locations and the blank password. (If you suspect someone else might get access to the machine you're using, you can set a password for the key.)

Now enter **ssh-copy-id** followed by the hostname or IP address of the server; your public key will be transferred over to that server. Try logging in and you should see that you don't need to specify a password any more. If it all works, edit **/etc/ssh/sshd_config**, change the **PasswordAuthentication** line to **no**, and restart OpenSSH. (And never give away your private key – it's **~/.ssh/id_rsa**!)

```

auth.log (/var/log) - VIM
File Edit Tabs Help
385 Apr 27 19:16:46 okachi sshd[23764]: input userauth request: invalid user root [preauth]
386 Apr 27 19:16:46 okachi sshd[23764]: pam_unix(sshd:auth): authentication failure; logname= u
id=0 euid=0 tty=ssh ruser= rhost=116.10.191.190 user=root
387 Apr 27 19:16:48 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
388 Apr 27 19:16:51 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
389 Apr 27 19:16:54 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
390 Apr 27 19:16:56 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
391 Apr 27 19:16:59 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
392 Apr 27 19:17:01 okachi CRON[23768]: pam_unix(cron:session): session opened for user root by
(uid=0)
393 Apr 27 19:17:01 okachi CRON[23768]: pam_unix(cron:session): session closed for user root
394 Apr 27 19:17:02 okachi sshd[23764]: Failed password for invalid user root from 116.10.191.1
90 port 2173 ssh2
395 Apr 27 19:17:02 okachi sshd[23764]: Disconnecting: Too many authentication failures for roo
t [preauth]
396 Apr 27 19:17:02 okachi sshd[23764]: PAM 5 more authentication failures; logname=uid=0 euid
=0 tty=ssh ruser= rhost=116.10.191.190 user=root
397 Apr 27 19:17:02 okachi sshd[23764]: PAM service(sshd) ignoring max retries; 6 > 3
398 Apr 27 19:17:06 okachi sshd[23775]: User root from 116.10.191.190 not allowed because not l
isted in AllowUsers
399 Apr 27 19:17:06 okachi sshd[23775]: input userauth request: invalid user root [preauth]
400 Apr 27 19:17:07 okachi sshd[23775]: pam_unix(sshd:auth): authentication failure; logname= u
id=0 euid=0 tty=ssh ruser= rhost=116.10.191.190 user=root
401 Apr 27 19:17:09 okachi sshd[23775]: Failed password for invalid user root from 116.10.191.1
90 port 2193 ssh2
auth.log 400,1 0%

```

Here's **/var/log/auth.log** (again with lovely Vim syntax highlighting) on a sample server, with the red lines showing root login attempts by bots.

One enormously useful add-on for OpenSSH is Fail2ban. This is a program that monitors unsuccessful login attempts; if a certain IP address fails to log in too many times, that IP is automatically blacklisted. This again adds more work for crackers and bots, as they can't keep trying to log in from the same IP address and need to switch periodically.

On Debian it's a simple **apt-get install fail2ban** away, and it starts up automatically.

By default it automatically blocks IPs (using the system's **iptables** command) for 600 seconds if they have six failed login attempts. You may want to raise the duration to something much longer, and also allow IPs a few more attempts – you don't want to make a few typos when entering your password and accidentally ban yourself!

Fail2ban's main configuration file is **/etc/fail2ban/jail.conf**. However, it's a bad idea to edit that directly (as your changes could be overwritten by system updates), so copy it to **/etc/fail2ban/jail.local** and edit that file instead. The **bantime** and **maxretry** options towards the top control the default settings we mentioned before, and you can also exempt certain IPs from being banned in the **ignoreip** line.

But hang on – **maxretry** here at the top has a value of three, yet we mentioned earlier that there must be six failed login attempts for Fail2ban to take effect! This is because there's a special “[ssh]” section further down that overrides the default settings. You'll see that Fail2ban can be used with other services than SSH too. Once you've made your changes, restart the program like so:

```
service fail2ban restart
```

“The default OpenSSH configuration means that bots don't have to do much work.”

2 HARDENING APACHE

The standard Apache web server configuration in Debian is fairly secure and usable out of the box, but can be made even tighter by disabling a few features. For instance, try to access a non-existing URL in your Apache installation, and at the bottom of the “404 not found” screen that appears you’ll see a line like this:

Apache 2.2.22 (Debian) Server...

It’s best not to tell the world the exact version of Apache you’re using. Vulnerabilities that affect specific versions occasionally appear, so it’s best to leave crackers in the dark about your exact setup. Similarly, Apache includes version information in its HTTP headers: try **telnet <hostname> 80** and then **HEAD / HTTP/1.0** (hit Enter twice). You’ll see various bits of information, as in the screenshot.

To disable these features, edit the Apache configuration file; in many distros this is **/etc/apache2/apache2.conf**, but in the case of Debian, its security-related settings are stored in **/etc/apache2/conf.d/security**, so edit that instead. Find the **ServerSignature** line and change **On** to **Off**, and then find the **ServerTokens** line and make sure it’s just followed by **Prod** (ie the server will just say that it’s the Apache “product”, and not give out specific version information). After you’ve made the changes, restart Apache with:

service apache2 restart

Apache also tries to be helpful by providing directory listings for directories that don’t contain an **index.html** file. This feature, provided by the Apache module **autoindex**, could be abused by hackers to poke around in your system, so you can disable it with:

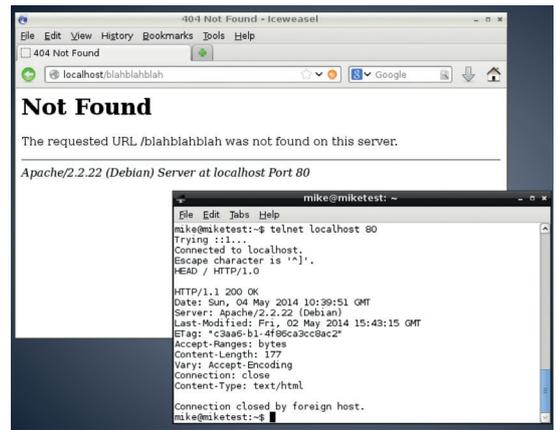
a2dismod autoindex

Status report

Another initially helpful (but risky on production machines) module is **status**: this lets you go to **http://<hostname>/server-status** and get a bunch of information about the configuration and performance. In Debian it’s only possible to access this page from the same machine on which Apache is running, but this may vary in other distros, so it’s wise to turn it off unless you really need it using **a2dismod status**.

There’s a very useful module called ModSecurity, which you can grab with a quick:

apt-get install libapache2-modsecurity



Apache is telling the world its exact version details, both in 404 pages and HTTP headers – but we can fix that.

This is an exceptionally powerful module that can protect against SQL injection attacks, cross-site scripting, session hijacking, trojans and other risks. After installation a configuration file is placed in **/etc/modsecurity/modsecurity.conf-recommended**; rename this and remove the **-recommended** part to activate it. The rules for detecting attacks are provided in **/usr/share/modsecurity-crs/** – go there and have a look inside the **base_rules**, **optional_rules** and **experimental_rules** directory. Each **.conf** file inside has some comment text explaining what it does, so if you find something useful, copy (or symlink) it into the **/usr/share/modsecurity-crs/activated_rules** folder.

Next, you’ll need to tell ModSecurity to use these rules. Edit **/etc/apache2/mods-enabled/modsecurity.conf** and beneath the **Include "/etc/modsecurity/*conf"** line, add these lines:

Include "/usr/share/modsecurity-crs/*conf"

Include "/usr/share/modsecurity-crs/activated_rules/*conf"

Now restart Apache to activate the configuration. By default, ModSecurity only detects problems and doesn’t act on them, logging its work to **/var/log/apache2/modsec_audit.log**. This gives you time to see how the rules will affect your site (and if they could break anything). When you’re confident with everything, make ModSecurity actively prevent exploits by opening **/etc/modsecurity/modsecurity.conf** and changing the **SecRuleEngine** option from **DetectionOnly** to **On**. Finally, restart Apache.

3 HARDENING YOUR SYSTEM

So that’s two of the most commonly used server programs hardened: OpenSSH and Apache. What you do from here depends on your particular setup, eg whether your server will primarily be used for email or databases. Still, there are many other things you can do to enhance the general security of your Linux installation. It’s a good idea to use an IDS, for instance

– an Intrusion Detection System, which keeps an eye on critical system files and alerts you if they change. This is a good way to see if someone has gained remote access to one of your machines and is tampering with configuration files.

Another useful program is an auditing tool. There’s a good one in Debian’s package repositories, called

LV PRO TIP

ModSecurity is loaded with advanced features, so visit www.modsecurity.org/documentation for all the details.

Tiger, and although it hasn't been updated for a while, it's still useful for finding holes in your setup. Run:

apt-get install tiger

Doing this will also install Tripwire, the IDS we'll be using. Once the packages have been downloaded you'll be prompted for two passwords; these are used to protect two keys that will be used to protect configuration files (after all, auditing and file checking tools aren't much use if they can also be easily exploited). Enter something memorable, and once the configuration has finished, enter:

tiger -H

This will start an extensive security scan of the system, and might take a few minutes depending on the speed of your machine. (Don't be alarmed if your hard drive thrashes a lot during this procedure!) At the end, Tiger will generate a HTML file and show you exactly where it is stored in `/var/log/tiger/`. Open it up (you could use the brilliant text-mode Elinks browser if you're logged in via SSH) to get a comprehensive report that lists potential risks in your system.

These include: file permission problems; processes listening on network sockets; poor configuration file settings; accounts without valid shells; and more. Tiger uses checksums to see if system files have changed after their initial installation, so if an intruder puts a trojan in a binary in `/sbin`, for instance, Tiger will tell you in the report that it differs from the original packaged version.

Every warning is accompanied by a code such as `acc022w`. To get a detailed description of the warning, enter this as root:

tigexp acc022w

It's very helpful, as it often suggests fixes as well. See the manual page for Tiger (`man tiger`) for other report formats and extra options.

Advanced file checking

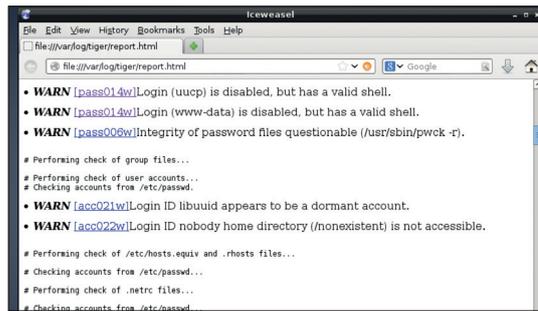
While Tiger is useful for checking executables against their original packaged versions, Tripwire goes a lot further and lets you spot changes all over the filesystem. To set it up, enter:

tripwire --init

This creates a database of file information that will be used when you perform a check. (You may be prompted for one of the passwords you specified

```
File Edit Tabs Help
-----
Report Summary:
-----
Host name:          miketest
Host IP address:    127.0.1.1
Host ID:           None
Policy file used:   /etc/tripwire/tw.pol
Configuration file used: /etc/tripwire/tw.cfg
Database file used: /var/lib/tripwire/miketest.twd
Command line used:  tripwire --check
-----
Rule Summary:
-----
Section: Unix File System
-----
Rule Name          Severity Level  Added  Removed  Modified
-----
Other binaries     66              0      0         0
Tripwire Binaries  100             0      0         0
Other Libraries    66              0      0         0
Root file-system executables 100             0      0         0
Tripwire Data Files 100             0      0         0
```

Tripwire can monitor any directory on your system, and give you an instant report listing any files that have changed.



Tiger gives a good overview of potential security flaws in your setup, and the **tigexp** tool provides more detailed descriptions.

when you installed Tiger earlier.) To see that the database works, edit a file in `/etc` – you could add a comment to `/etc/rc.local` for instance. Then run:

tripwire --check > report.txt

Now look in `report.txt` and do a search for `"rc.local"` (or the file you changed). You'll see something like this:

Modified:

"/etc/rc.local"

Nice and simple – it tells you exactly which files have changed. At the start of the report you'll see a useful summary as well. There's one problem in the default setup, though: Tripwire monitors `/proc`, and as that's constantly changing (because it contains information about running processes), it clogs up the report with unimportant text. To fix this, we need to change the Tripwire policy that defines which directories it should monitor. Edit `/etc/tripwire/twpol.txt` and find this line:

/proc -> \$(Device) ;

Delete this line and enter the following to update the policy database:

twadmin --create-polfile /etc/tripwire/twpol.txt

Now we need to rebuild the filesystem database, so go into the `/var/lib/tripwire` directory and remove the `.twd` file contained therein. Run `tripwire --init` and generate a report, and you'll see that `/proc` is no longer included in the report.

Have a more detailed look inside `/etc/tripwire/twpol.txt` to see what Tripwire can do, including different types of warnings for different directories. If you make a change to a system file and don't want Tripwire complaining in every report, you'll need to update the database. In `/var/lib/tripwire/report`, find the most recent report (eg with `ls -l`). Then run:

tripwire --update --twrfile <report>

Replace `<report>` here with the most recent version. The report will open in a text editor, and as you scroll down, you'll see changed files listed like so:

[x] "/etc/rc.local"

This means that the file is selected for updating in the database, so you won't be warned about it next time. (If you still want to be warned about that file, remove the **x**.) Save the file and exit the editor, and after the next `--check` command you'll see that the complaint is gone. 

Mike Saunders is the author of *The Haynes Linux Manual*, writer of the MikeOS assembly language operating system and has been messing with Linux since 1998.

LV PRO TIP

If you'd like us to run a separate tutorial on hardening another piece of server software, drop us a line at letters@linuxvoice.com.

VIRTUALBOX: CONVERT AN XP BOX INTO A VIRTUAL MACHINE

MARK CRUTCH

Ease the move to Linux by bringing your old Windows XP machine with you.

WHY DO THIS?

- Keep hold of your old XP installation
- Save £5.5m in support costs
- Move to Linux without the risk of losing XP functionality

On 8 April this year Microsoft issued its last update for Windows XP, leaving it vulnerable to future security exploits. Despite months of advance warning there are still millions of machines running this now obsolete operating system. Although this may seem like a perfect opportunity to migrate to Linux, many users are still reticent to give up on their old machines. One way to overcome this inertia is to convert the old Windows box into a virtual machine (VM) that will run inside the new Linux system, providing the reluctant user with a digital security blanket to cling to.

The real aim is to reduce the number of XP systems that are connected to the internet: every machine taken offline is one less that can host malware or participate in a denial of service attack. With that in mind we'll not only convert the physical box to a virtual one, but also include a few tips to ease the move to Linux and reduce the need to boot the Windows VM or put it online.

We'll be migrating a Windows XP machine, but the same approach also works with Vista and Windows 7. Due to hardware differences, licensing rules and various OEM flavours of Windows, not every machine will migrate using this approach – but we've had far more successes than failures. Although our

destination machine is a Linux box, you can use this same method to migrate your old system to a virtual machine running on a MacOS host, or even another version of Windows if you really want to.

Gather your hardware

Before starting our migration, it's worth noting a few hardware requirements. Virtual machines can quickly eat into available memory, drive space and processing power, so a capable host machine is a must. Take a look at the old XP machine to determine how much memory it has, and how much of the hard drive is in use, then ensure that the host has sufficient spare capacity to cover both the virtual machine and its own day to day usage. A large USB hard drive will also make things a bit easier, although it's not essential as long as you can move large files around using a network connection.

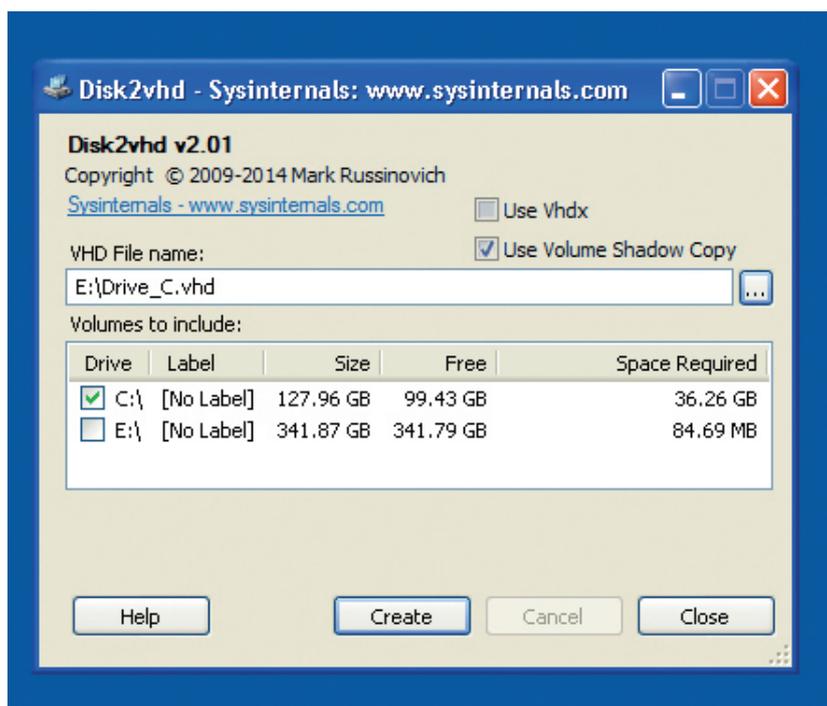
The Windows licence is probably tied to the old physical machine, so strictly speaking you should keep the hardware for as long as you have the virtual machine. You'll probably need the Windows Product Key from the sticker on the old box, but if that's too faded to read there are programs available that can extract the product key from a running Windows system. Although Microsoft's anti-piracy restrictions can sometimes cause problems, most XP machines migrate relatively smoothly. There are no guarantees, though, especially with XP Home and OEM installations, so don't go spending lots of money on extra RAM and a bigger hard drive until you're sure the migration will be a success.

Clone your hard drive

We'll start the migration by creating a disk image that we can use directly in VirtualBox. There are a variety of ways to do this, but for this tutorial we'll be using Disk2vhd on the Windows box. This can be downloaded free of charge (<http://bit.ly/18b901i>), but remember that we're trying to keep the XP machine off the internet, so it's probably best to download the file using another machine and then copy it to the XP box via a USB drive. Unzip the file, enter the directory, and double-click on the EXE file to run it.

Once you've accepted the EULA you'll be presented with the Disk2vhd dialog. The controls always seem to be in the wrong order, in our opinion, and we usually approach them from bottom to top. First, therefore,

Disk2vhd was written to create disk images for Microsoft's own VirtualPC program, but it works just as well with VirtualBox.



is the Volumes To Include panel, which lists all the drive partitions that XP knows about. Ensure that the partitions on the internal drive are checked, and that any partitions on the USB drive are unchecked. If you have multiple physical drives in the machine it's probably best to export each of them separately – all the partitions for the first drive into one file in the first pass, then run the program again to export all the partitions for the second drive into another.

Moving up the dialog we get to the VHD Filename. Use the button on the right to browse to your USB disk. If necessary you can use the internal drive for the file destination, but performance will suffer, and you'll need a lot of free space. Finally, confirm the state of the two checkboxes at the top of the dialog: we want to create a plain VHD file, so uncheck the Use Vhdx option; we do, however, want to check the Use Volume Shadow Copy option, which utilises a feature built into XP and later versions of Windows to snapshot the hard drive for imaging. This is especially vital if you're creating the file on the source drive.

With all the options set, it's time to click on the Create button and leave it working for a while. How long will depend on the amount of data and the speed of the machine and the drives, but it typically takes hours rather than minutes.

Prepare VirtualBox

While the XP box is being imaged, we can take the time to install and configure VirtualBox on the host. Most distros' repositories tend to lag some way behind the official release, so we'll download it directly from the VirtualBox website (https://www.virtualbox.org/wiki/Linux_Downloads). The top of the downloads page has links to DEB and RPM files, but if you scroll down a little you'll find instructions for installing from the VirtualBox repositories.

The core of VirtualBox is licensed under the GPL, but there's an additional commercial extension that you'll probably want to use. This is licensed under Oracle's own "PUEL" licence, which allows for personal, academic and evaluation use at no charge. Note that "personal use" includes using it in a commercial setting if you've installed it yourself, but do check the wording of the licence (https://www.virtualbox.org/wiki/VirtualBox_PUEL) if you're using it for anything other than inarguably personal or academic reasons.



The extension pack adds various features to the base VirtualBox system, the most notable being USB 2.0 support. If you're migrating XP to support a printer or other USB hardware that has poor Linux drivers this might be an important consideration – although VirtualBox's USB support isn't perfect, especially when dealing with esoteric drivers, so make sure you test the final system thoroughly.

Installing the extension pack is as simple as downloading it from the link on the VirtualBox download page (<https://www.virtualbox.org/wiki/Downloads>), then opening it with the main VirtualBox application. If your desktop has a suitable file association set up you'll probably be offered the option to open with VirtualBox when you download the file; otherwise you can manually add it via the File > Preferences > Extensions panel in the main VirtualBox manager.

With VirtualBox installed we're going to create a new VM. At this time it won't have a hard drive – that's probably still being imaged – but we can get the rest of the machine in place. Start by launching the VirtualBox manager and clicking the New button to bring up a wizard.

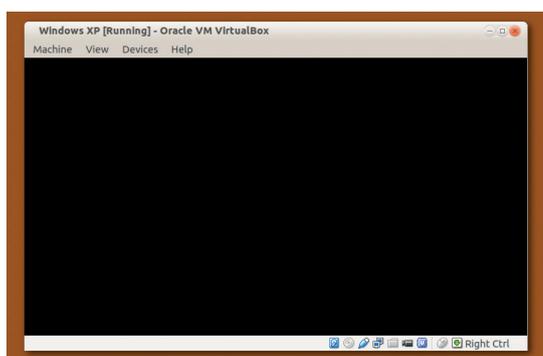
Give your VM a name: this will also be used as a directory name for holding your machine's files. Ensure that you pick the values for Type and Version that correspond to the machine you're migrating. In our case that's Microsoft Windows and Windows XP, respectively, but if your source machine is one of those rare beasts that's running the 64-bit version of XP you'll need to pick that specifically.

On the next page of the wizard you'll need to set the size of the virtual machine's memory. If you can

While you're at the VirtualBox website, make sure you also download the extensive user manual.

LV PRO TIP

To prevent XP connecting to the internet, uncheck the Cable Connected option in the VirtualBox network settings so that Windows simply thinks the Ethernet lead has become unplugged. That way, should you find you have to connect to the network in future, you can just re-check that option to reconnect the virtual cable.



Either your desktop is using a theme with black text on a black background, or you've got the wrong HAL.

Accessing your Windows files

If you want to copy files from your Windows machine to the Linux host, the simplest approach is probably to just attach a USB drive to Windows via the VirtualBox Devices menu, copy the files, detach it from VirtualBox to make it available to Linux and then copy the files back off it into the host.

That's fine for a one-off transfer, but for more frequent use you can set up a file share within XP using Windows' own SMB protocol, which can then be mounted on your Linux host. VirtualBox also has a Shared Folders pane in the VM's settings dialog that will let you share host folders with the Windows machine in a similar way, enabling you to "push" files to the Linux box from within XP. In our experience it's usually easier to share a Windows folder, then connect using Nautilus, Dolphin or Caja with the SMB protocol and the IP address

of the virtual machine – or use the lower-level Samba tools if your desktop environment doesn't understand the **SMB://** URL syntax.

One problem with all of these file transfer methods is that they require the Windows machine to be running. If all you want to do is get some files out of the Windows drive, there are various ways to mount the VM's disk image directly as a block device in the Linux filesystem. We've had most success with the **guestmount** command line tool, which mounts the drive using FUSE and is available in the repositories of most mainstream distributions. A word of warning: on our Linux Mint system we also had to install **libguestfs-tools** to get it working, which in turn pulls down a lot of files.

Using **guestmount** you can mount your Windows drive in read-only mode with the following

command as root (or prefix with **sudo** if your distribution needs it), replacing the VDI file and the **~/Windows** mount point as appropriate:

```
guestmount -a Windows_XP.vdi -i --ro -o allow_other ~/Windows
```

To unmount the disk image when you're finished with it, use:

```
fusermount -u ~/Windows
```

Although these are run as root, the **-o allow_other** FUSE option lets any user access the files, so you can copy files out of the Windows drive and into the Linux environment as a regular user. Despite both the VDI file and the mount point being owned by our normal user account, we have to use **sudo** on our Mint box for this approach to work. If anyone has any tips on getting **guestmount** to work as a regular user, please post them to the Linux Voice forums.

spare it, allocate the same amount as is present in the physical source machine. The third page is where we'll tell VirtualBox that we don't want a hard drive, and then finally create our new machine – but not until VirtualBox has offered a final warning about our lack of a disk.

Congratulations: you now have a half-imaged physical machine and a disk-deprived virtual machine. Take a well-earned break while the imaging chugs its way to completion.

Add the virtual hard drive.

With the imaging process over, you should now find a large VHD file on the USB drive. Copy any other files that you want off the XP box, then finally shut down the physical machine forever (hopefully). Mount the USB drive on the host machine, and copy the VHD file into your new virtual machine's directory. Unless you specifically chose otherwise it's probably in a folder called VirtualBox VMs in your home directory.

Return to the VirtualBox manager and select your XP VM in the list on the left. Click the heading of the Storage section in the right-hand panel to open the VM's settings dialog with the storage page showing. Select the IDE controller and click on the small icon

to add a hard disk (check the tooltips to distinguish between the two small icons, if it's not clear which one represents a hard disk). In the resultant dialog you should select Choose Existing Disk, then pick your VHD file from the VM's directory.

We're close to starting our virtual machine, but it's worth taking a couple of minutes to step through the other settings pages. If you've installed the VirtualBox extension pack then it's worth checking that USB 2.0 is enabled. We also like to enable the Remote Display option, which lets you access the screen of your running VM from another machine using a remote desktop client such as Remmina or Rdesktop. On the Advanced tab of the General Settings panel it's usually worth enabling the Shared clipboard feature. Set it to bidirectional to let you copy and paste text between the Linux host and the virtual machine. We tend to use the Description tab in the same panel to hold a copy of the Windows Product Key, to save me rummaging around the loft for the physical box if I need to enter it in future.

One final thing to set up is networking. We don't really want this machine on the internet, but you'll probably want it connected during its first boot to deal with Windows' activation requirements. On the Network panel, enable the first adaptor, and choose NAT from the first pop-up menu. Ensure the Cable Connected setting is checked.

With all that done it's time for the big moment. Close the settings dialog, ensure the XP VM is selected on the left, click the Start button in the VirtualBox toolbar, and watch your new virtual machine boot...

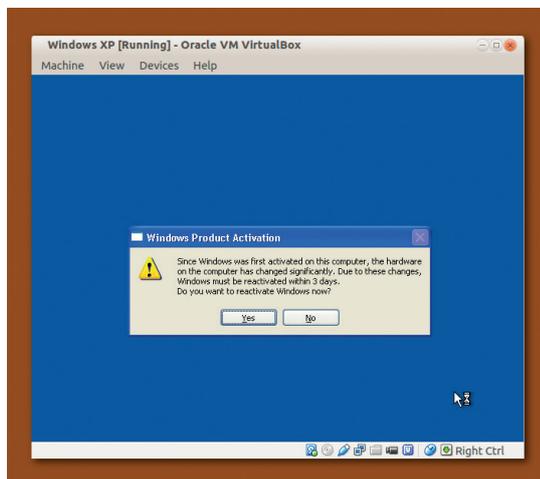
I'm sorry, Dave. I'm afraid I can't do that.

If you're very lucky you're now sitting in front of a VirtualBox window showing the XP login screen, or a Windows activation screen. More probably you're looking at a black window, with the icons in the VirtualBox status bar showing no disk or network activity. You've just been halted by HAL.

LV PRO TIP

Once the main migration is working you may wish to clone your VHD file into VirtualBox's native VDI format, as it's likely that the code for its own format is better tested and more robust. Select the File > Virtual Media Manager menu in the VirtualBox Manager, then release your VHD file from the VM. Copy it to a dynamically allocated VDI file, then attach it to the Windows machine via the Settings dialog. Check that it works, then delete the VHD file.

You've changed your hardware. Pirates sometimes change their hardware. Ergo, you are a pirate until you prove otherwise



JOHN VON NEUMANN, EDVAC, AND THE IAS MACHINE

The Linux Voice time machine takes us back to one of computing's eureka moments: the von Neumann architecture.

John von Neumann was born in Hungary in 1903. He was a prodigy, publishing two major mathematical papers by the age of 19. After teaching at the University of Berlin, in 1930 he was invited to Princeton University in the US, and later joined the Institute for Advanced Study there. During this time he contributed to several branches of maths, including set theory, game theory, quantum mechanics and logic and mathematical economics.

During the late 1930s, he worked on modelling explosions, which led to his involvement in the Manhattan Project. He is also credited with developing the strategy of "mutually assured destruction" which drove the Cold War. (In game theory, mutually assured destruction is an equilibrium, in which neither player has the incentive either to act or to disarm.)

Von Neumann was also heavily involved in early computing, partly because the work he was doing on the hydrogen bomb required vast and complex calculations. These were done initially by human computers – women using desk calculators to run the calculations required, on a production-line basis. During 1943 they began to use IBM punched-card machines, which worked at roughly the same speed but didn't need sleep. (A single calculation problem took three months, which Richard Feynmann reduced to three weeks by running cards in parallel on the machines.) These machines, however, weren't programmable computers; they were just calculators.

Von Neumann consulted on both the ENIAC and EDVAC projects. The initial design of the ENIAC, the first programmable general-purpose computer, did not include the ability to store programs, and while it was programmable and Turing-complete, the programming was done by manipulating switches and cables. (Colossus was programmed similarly, but was not general-purpose, being dedicated to cryptanalysis.) ENIAC used an immense number of vacuum tubes to both store numbers and calculate, and punch cards for input and output. It was developed to run artillery calculations, but due to the involvement of von Neumann and Los Alamos, in the end the first calculation it ran was computations for the hydrogen bomb, using around a million punch cards.

EDVAC and the First Draft of a Report

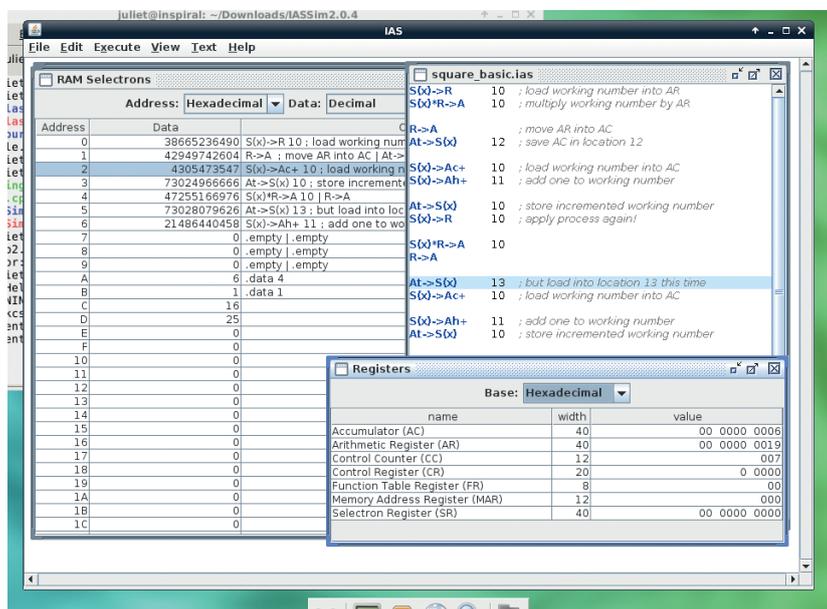
The EDVAC was proposed by the inventors of ENIAC, Mauchly and Eckert, in late 1944 – before the ENIAC was fully operational, but using improvements thought of while building it. EDVAC, like ENIAC, was built for the US Army's Ballistics Research Laboratory (at the Aberdeen Proving Ground). Although it hadn't yet been built, von Neumann's famous First Draft of a Report on the EDVAC was written (by hand while commuting to Los Alamos by train) in June 1945.

The Draft Report contains the first published description of the logical design of a stored-program computer, specifically the design that is often now known as the Von Neumann architecture and which is still widely used today. However, there is controversy over the extent to which this was solely von Neumann's work.

Some of the EDVAC team maintained that the concepts arose from discussions and work at the Moore School (where EDVAC was designed) before von Neumann began consulting there. Other documents suggest that Eckert and Mauchly had already thought of the idea of a 'stored program', but they hadn't fully outlined a design.

The First Draft of a Report on the EDVAC was, indeed, a first draft. It was intended as a summary and analysis of the logical design of the proposed EDVAC, with further extensions and suggestions from von Neumann. In it, von Neumann recommended that the computer have a central control unit to control all operations, a central processing unit to carry out operations, and a memory that could store programs and data and retrieve them from any point (ie random

To run the emulator on Linux and study von Neumann's programming methods, you will need Java version 5 or later.



access, not sequential access). He also recommended that EDVAC have a serial, rather than a parallel, processor, as he was concerned that a parallel processor would be too hard to build.

Unfortunately (and apparently without von Neumann's knowledge), Goldstine distributed the First Draft with just his and von Neumann's names on it, and without any credit given to Eckert and Mauchly. (From the gaps in the report, it is likely that von Neumann intended to insert further credits before 'proper' publication.) Goldstine likely only meant to share the ideas as quickly as possible, but it had the unfortunate effect of linking this architecture with von Neumann alone, rather than with the whole group of people who had been working on it.

When EDVAC was in due course built, it had a computational unit that operated on two numbers at a time then returned the results to memory, a dispatcher unit which connected this to the memory, three temporary operational tanks, nearly 6,000 vacuum tubes, and a mercury delay line memory of 1,000 words (later 1,024 words). It read in magnetic tape. It finally began operation in 1951, by which time von Neumann had moved back to IAS; not only that, but the Manchester Mark I team in the UK (who were later joined by Turing) had beaten them to the post of developing the first stored-program computer, running their machine for the first time in June 1948.

The IAS Machine

Meanwhile, in 1946, von Neumann wrote another paper, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", which further developed his ideas. The IAS machine was the embodiment of those ideas.

One of the big differences between the IAS machine and EDVAC was that the IAS machine had a parallel processor. Words were processed in series, but the bits in each word were stored and operated on in parallel. This shows how fast the technology was moving – in the report on EDVAC in 1945, von Neumann thought that a parallel processor would be too difficult to build, so recommended a serial processor. By the time IAS machine project started in May 1946 (or possibly soon after, while they were working on the design), von Neumann had become convinced that parallel processing could work.

The IAS machine itself used a 40-bit word (with two 20-bit instructions per word), with a 1024-word memory and two general-purpose registers. Unlike many other early computers, it was asynchronous, with no clock regulating instruction timing. Instructions were executed one after the other. It used vacuum tubes for its logic, and Williams tubes (cathode ray tubes) for its memory, known as the Selectron.

Cathode ray memory relies on the fact that when a dot is drawn on a cathode ray tube, the dot becomes positively charged and the area around it negatively charged. When the beam is next pointed at that location, a voltage pulse is generated, which will differ



Von Neumann invented cellular automata. Turing invented parts of mathematical biology. These days, cellular automata are at the forefront of our investigations into mathematical biology – and those investigations rely on the computers that Turing and von Neumann put so much work into.

depending on whether there was a 'dot' or a 'dash' stored there. A metal pickup plate over the tube detects the voltage pulse and passes the data out to the next part of the memory system and ultimately to the control unit. The act of reading the memory also wipes it, so it must immediately be rewritten; and as the charge well fades quickly, the whole thing must also be frequently rewritten. The advantage, though, over mercury delay lines was that as the beam could point at any location immediately, memory was entirely random-access. With mercury lines, you had to wait until your data word came around to the output of the line before you could read it.

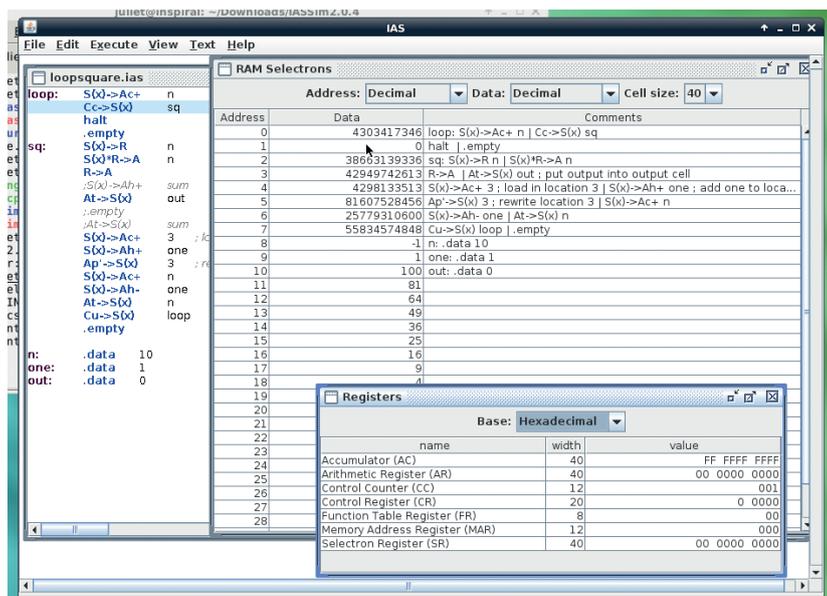
Von Neumann architecture

The crucial point about the 'von Neumann' architecture was that it combined both instructions and data in a single memory. This meant that you could, for example, implement a loop by modifying stored instructions. Unfortunately this also has the effect that all operations are using the same memory, so the machine cannot fetch an instruction and act on data at the same time. This came to be known as the von Neumann Bottleneck. The alternative, the Harvard Architecture (originating with the Harvard Mark I), separates data and instruction storage. Most modern computers use von Neumann architecture for main memory, but a modified Harvard architecture is used for some caches and in some other situations.

The IAS had five main parts: Central Arithmetic (which performed arithmetic operations), Central Control (which passed information between CA and memory), Memory, Output, Input, and the recording medium (magnetic tape, initially). It had seven registers, three in the CA and four in CC:

Central Arithmetic

- AC Accumulator.
- MQ/AR Multiplier/Quotient register (aka Arithmetic Register).



Using a little more assembly language makes coding loops straightforward.

- MDR Memory Data Register.
- Central Control
- IBR Instruction Buffer Register.
- IR Instruction Register.
- PC Program Counter.
- MAR Memory Address Register.

The IAS instructions took the form of “8-bit operation code” + “12-bit memory address” (the memory address was ignored if the instruction did not need it). So the instruction **S(x)->R 010** meant “load the number at Selectron location **x** into the Arithmetic Register; location **x** is **010**”. The available instruction set had 21 operations (plus Halt making 22), which copied numbers into and out of the AC and AR, subtracted, added, multiplied, or divided them, and controlled execution. The execution control enabled the programmer to jump to a particular memory address, or to check whether a given value was greater than or equal to 0, or to rewrite a given instruction; it was these abilities that enabled loops.

The IAS architecture and plans were implemented in several machines across the world, as the plans were freely distributed. However, all of these machines, although IAS derivatives, were slightly different; you couldn't just run software written for one machine on another machine without rewriting it for the quirks of that individual machine. Some of the famous IAS machines include MANIAC (at the Los Alamos National Laboratory; von Neumann was involved with this one too and was responsible for the name), the IBM 701 (IBM's first commercial scientific computer, with 17 installations), and ORACLE (in Oak Ridge National Laboratory). Other IAS machines existed in Copenhagen, Moscow, Stockholm, and Sydney, among others.

IAS emulator

There's a Java-based emulator, IASSim, available for the Princeton IAS machine, from www.cs.colby.edu/djkskrien/IASSim/ – so you can try out IAS machine

coding for yourself. Download the Zip file, unpack it, and **cd** into the folder. This command will launch the emulator in its own window:

```
java -cp IASSim2.0.4.jar:jhall.jar:IASSimHelp2.0.jar iassim.Main -m
```

IAS.cpu

You can load in an assembly language text file from the File menu, and there is a tutorial and online help available from the Help menu.

The folk who wrote the emulator have also written a basic assembler for it, to make life a little easier. For this first example I'll use the assembler as little as possible, to give you the best flavour of the IAS machine's language. Open up a new text file in the emulator and enter this (without the line numbers):

```
0. S(x)->R 10 ; load working number into AR
   S(x)*R->A 10 ; multiply working number by AR
1. R->A ; move AR into AC
   At->S(x) 12 ; save AC in location 12
2. S(x)->Ac+ 10 ; load working number into AC
   S(x)->Ah+ 11 ; add one to working number
3. At->S(x) 10 ; store incremented working number
   S(x)->R 10 ; and start again!
4. S(x)*R->A 10
   R->A
5. At->S(x) 13 ; but save in location 13 this time
   S(x)->Ac+ 10
6. S(x)->Ah+ 11
   At->S(x) 10
7. .empty
   .empty
8. .empty
   .empty
9. .empty
   .empty
10. .data 4
11. .data 1
```

Let's take a look at that. First of all, each 'line' (which is in fact the register address where the instruction is stored) has two instructions, since the IAS machine had two instructions per 'word' on its tapes.

Line 0: The first half of our first pair of instructions loads the number in location 10 into **AR**, the Arithmetic Register. **S(x)** refers to Selectron (memory) location **x**, and 10 is given for **x** at the end of the line.

The second half, **S(x)*R->A 10** multiplies **S(10)** by **R**, and stores the result in **A**. Multiplication on the IAS gave rise to a result stored in two halves: the left half of the number in **AC** and the right half in **AR**. Since we are only multiplying small numbers, only the right half is useful.

Line 1: The next instruction, **R->A**, therefore moves the right half of the result from **AR** into **AC**. We can then save it to location 12 with **At->S(x) 12**. That gives us the first answer, the square of the (working) number, stored in location 12.

Line 2: Load the working number itself into **AC**, then add the contents of location 11 to it (**S(x)->Ah+ 11**). As you'll see in a moment, location 11 contains 1, so this just increments our working number by 1.

Line 3: We store the incremented working number back in location 10, and start the process again.

Lines 4–6: As above, but this time around our new result is stored in location 13. We increment the working number one more time before stopping.

Lines 7–9: These are empty just for ease of setup. The empty lines mean that we can store our working number in location 10 and leave a bit of room to add more instructions if desired. (If you remember coding in BASIC with line numbers, you may recall numbering in tens to give yourself wiggle room; same thing!).

Lines 10-11: The assembler instruction .data is used to put the numbers 4 (our working number) and 1 (for use when incrementing) into locations 10 and 11. The original programmers would have just been able to write numbers (whether all zeros for an empty line, or data numbers) straight to tape.

To run this, go to the Execute menu and choose Clear, Assemble, Load, and Run. Check out the RAM Selectrons window to see the contents of the registers – you should see 16 in location C (hexadecimal) and 25 in location D. (You might need to change the Data view to Decimal.) You can also step through the program one instruction at a time using Debug mode, and watch the registers change in the Registers window, if you prefer.

In fact, if you change the Data view of the RAM Selectrons window to Hexadecimal, you can code your instructions directly into the Selectron locations. Each location has two sets of one 2-place and one 3-place hex number, corresponding to an instruction+location, twice. So, for example, the hex representation of **S(x)-R** is 09, and the first instruction of our first line is **09 00A** (A being 10 in hex).

Here’s a loop version of our squares code using assembly language (with thanks to the writers of the IAS Sim software for the loop control code):

```
loop: S(x)->Ac+ n ; load n into AC
      Cc->S(x) sq ; if n >= 0, go to sq
      halt
      .empty

sq: S(x)->R n
    S(x)*R->A n
    R->A
    At->S(x) out
    S(x)->Ac+ 3
    S(x)->Ah+ one
    Ap'->S(x) 3
    S(x)->Ac+ n
    S(x)->Ah- one
    At->S(x) n
    Cu->S(x) loop
    .empty

n: .data 10
one: .data 1
out: .data 0
```

The labels here are part of the assembly language, to make looping easier (but it could be done by hand if

you prefer – feel free to try it out!). We start off with **n** (see the data labels at the bottom), load it into the AC, and check that it is still non-negative. If so, we jump to the **sq** subroutine.

The first four lines of **sq** are familiar – load up **n**, square it, and store the square in the out location. Next is the interesting part.

S(x)->Ac+ 3 loads the instruction at location 3 into the AC register. Location 3, if you count lines (remember that the locations start at 0) contains the instruction **R->A** on its left side and **At->S(x)** out on its right side. So we now have a number representing those instructions in the AC.

The next instruction, **S(x)->Ah+** adds one to that. This effectively alters **At->S(x)** out to **At->S(x) out+1**.

We then write this altered instruction back to location 3, with

Ap'->S(x) 3 (specifically, **Ap'** alters the right-hand side of the instruction at location 3, and **Ap** alters the left-hand side). So the next time

we loop around this code, instead of writing the output to the location labelled out, we'll write it to the next location along. To watch this happen, you can use Debug mode, step through the code, and keep a close eye on the Registers window.

The next three lines load **n** up again, decrease it by one, and save it. We then jump back to loop with the **Cu** instruction, and go round the loop again.

If you load and run this, you'll see that you get the squares from 100–0 output in locations 10–20. This is the behaviour that the von Neumann architecture makes possible: altering the program’s stored instructions as you go along.

“Von Neumann designed a self-replicating computer program, which is considered to be the first computer virus.”

Final years

Von Neumann carried on working on computing, alongside his other areas of interest, for the rest of his life. In 1949, he designed a self-reproducing computer program, which is considered to be the first ever computer virus, and he worked on cellular automata and other aspects of self-replicating machines. He also introduced the idea of stochastic computing (which, broadly, uses probability rather than arithmetic) in a 1953 paper, although the computers of the time weren't able to implement the theory.

Sadly, he died in 1955 from bone or pancreatic cancer. (A biographer has speculated that this might have been due to his presence at the Bikini Atoll nuclear tests in 1946.) His contribution across his fields of interest was truly immense and he might well have contributed still further had he lived longer. 

Juliet Kemp is a scary polymath, and is the author of O'Reilly's *Linux System Administration Recipes*.

BEN EVERARD

FPGAs: BUILD YOUR OWN CUSTOM SOUND CHIP

Mine Bitcoins or make silly noises: anything's possible when you have the power to design your own chips.

WHY DO THIS?

- Make funky noises
- Create hardware customised for your exact needs
- Peek into the innards of chips and learn about IC design

Field Programmable Gate Arrays (FPGAs) are chips that can be programmed to perform different actions. However, they're programmable in a different way to CPUs and microcontrollers. An FPGA contains a series of logic circuits that can be programmed to connect to each other in different ways, so when programming an FPGA, you don't describe a series of steps like you would in Python or C but the circuit that you want to implement. In many ways, you can think of an FPGA as being a bit like a breadboard for prototyping your own chips. Because of this, you can program FPGAs to perform some tasks much quicker than they could be done on a CPU (such as mine bitcoins).

One common use of FPGAs is to re-implement processors and other chips from old computers such as games consoles. It's important to remember that

the FPGA doesn't emulate the design of a chip like a games console emulator may when running on a normal CPU: it actually implements it in

hardware. This has a massive effect on the speed at which it performs, because all the parts of the circuit are running in parallel not being simulated one at a time like they would be on a CPU.

There are a few FPGAs available, and a number of different prototyping boards for working them into your projects. Some need specialist equipment to program, while others can simply be plugged into a

USB device. In this tutorial, we're going to use a Papilio One 500K, which we bought from the Gadget Factory. In many ways, you can think of this as the Arduino of FPGAs. It's not the most powerful chip available, but it's enough to get started, and there are some easy-to-use tools to help beginners get started on it. The Gadget Factory also sells the Papilio Pro, which is more powerful than the model we're using (in FPGA terms, a device is more powerful if it has more components inside it, thereby enabling you to create larger and more complex circuits).

Gadget Factory also produces a range of wings, which are add-on hardware to enable things like audio and VGA output. Since the FPGA can be programmed to include the driver circuits for these, they're far simpler than they would be for microcontroller boards.

The language of hardware

The two common languages used to program FPGAs are Verilog and VHDL, but diving straight into the code isn't the easiest way to get started with the Papilio. Instead, you can start with circuits that other people have built. There aren't huge numbers of these, but there are a few. One of the most useful for Papilio users is the Zylin CPU (ZPU).

This core is also openly licensed (under a BSD-style licence). So, while running open source code on a ZPU, you're using fully open software on a fully open processing unit on a fully open board. There's enough freedom there to keep anyone's inner Stallman happy.

ZPUs aren't powerful enough to run a regular computer, but in microcontroller terms, the ZPU performs quite well. It's a 32-bit processor that runs at 96MHz. That makes it quite a bit faster than most Arduinos (about the same processing power as the Arduino Due). This means it's useful for embedded applications where you don't need a full OS stack, just a bit of processing power to control inputs and outputs. One of the most popular uses of the ZPU on the Papilio has been in making music synthesisers. The ZPU can control what's going on, while additional peripherals can generate the noises. That's what we're going to look at here.

GadgetFactory has modified the Arduino IDE to work with the ZPU and the Papilio One FPGA. We'll use that for our projects in this tutorial, so you'll need to get it from <http://forum.gadgetfactory.net/index.php?/files/file/8-zap-zpuino-arduino-papilio-ide>. Installing this is simply a case of unzipping it. Inside

“One common use of FPGAs is to re-implement processors from old computers.”



The Papilio FPGA board is one of the easiest ways to get started with FPGAs.

there's an executable called **zap**, which will start the IDE. If you've used the Arduino IDE before, you can use Zap in exactly the same way, but it does have a few features that are specially for the Papilio FPGA. You may have noticed that there's an extra menu called Papilio, and it's here that we'll get started.

Create a new project (known as a sketch in the Arduino terminology). Go to Papilio > New Papilio Project, and give it a name.

The programming language used is a dialect of C++. The two main functions are **setup()** and **loop()**. The **setup()** function is run when the Papilio first starts, then **loop()** runs continuously. In the new project you've just created, these will contain the code to simply turn pin 0 off and on a few times. Actually, the default code contains a typo: it uses a variable that's not declared. To correct it, change the loop function to:

```
void loop() {
  digitalWrite(0, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(0, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

Here, in the two places where the **led** variable was used, we've just put in the value 0 instead. This simple function sets pin 0 to **high**, then waits a second (the **delay** function takes the number of milliseconds as it's argument), then puts the pin low, waits another second and repeats.

This code isn't for the FPGA, remember, but for the ZPU processor. So, before you can use this code, you have to load a BIT file containing the processor onto the board. To do this, click on the following link, which should be in the comments at the top of the page:

[sketchdir://500K/papilio_one_500k.bit](#)

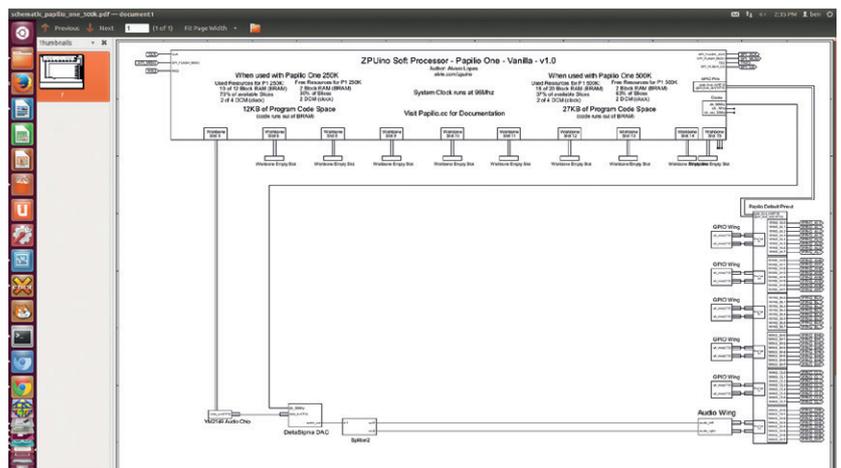
This will write a ZPU processor to the FPGA. Once it's finished, it'll display the following line in the console

Expanding the Papilio

There are three official 'MegaWings' made by the Gadget Factory to add features to the Papilio (all are compatible with all versions of the board).

- The LogicStart MegaWing adds a four-character seven-segment display, a VGA port, audio jack, 12-bit eight-channel SPI ADC, five direction micro joystick, eight LEDs and eight slide switches.
- The Arcade MegaWing adds a VGA port, a sound jack, PS/2 ports and two DB 9 joystick ports (for Commodore and Atari joysticks).
- The RetroCade MegaWing adds two stereo audio jacks, MIDI in, out and through, microSD card, micro joystick, 2x16 character LCD display, 16 analogue inputs and 16 digital inputs.

The first of these is designed for people who want to explore the general possibilities of the Papilio while the second and third provide functions for Arcade machines and music synths respectively. There are also a wide range of single-function wings available. Head to www.gadgetfactory.net/papilio to see all the options.



at the bottom of the ZAP window:

```
DONE = 0
```

Once this is finished, you can upload your code to the processor by clicking on the arrow icon in the top-left corner of the window.

There won't be any instant sign that it's worked, so you need to connect an LED to the appropriate pin along with a resistor to stop it drawing too much current. Any colour LED will do, but it needs to be connected the right way around (one of the legs will be a little shorter and there will be a flat side on the base. This is the negative leg). To stop it drawing too much current, you'll also need a resistor (any value between 220 and 1,000 ohms will be fine).

First, unplug the Papilio. You need to connect the positive leg of the LED to pin AL_0 on the Papilio One. If you look at the board with the USB port on the top side, this is the pin in the middle of the three in the bottom-right corner. The negative leg of the LED should connect to the resistor, and then the resistor should connect to the ground. You'll need a breadboard or some other circuit-building hardware to link everything together.

Once all this is connected, plug the Papilio back in and the LED should start to blink on and off.

Chip tunes

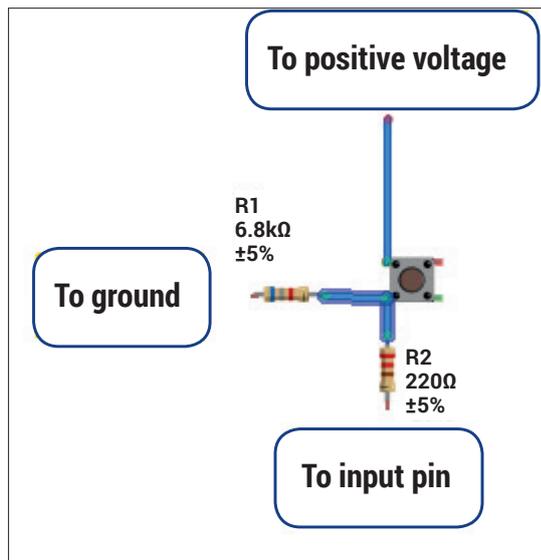
The ZPU in an FPGA is a way to get a slightly more powerful Arduino, but the Papilio can be far more than just a souped-up microcontroller. The ZPU doesn't take up all of the space in the Papilio One 500K. You can use the remaining space to implement additional features that you might find useful.

Go to Papilio > Papilio Examples > Audio YM2149 Simple in the Zap menu. This will open a new sketch that includes an implementation of a YM2149 audio chip. The music geeks among you will know this as the audio chip from the Atari ST (and several other games systems). This example can be used to play music files from these old systems to generate some retro sounds.

To see the schematic for the FPGA used in this sketch, click on [sketchdir://schematic_papilio_one_500k.pdf](#). You'll see that this contains three

All of the Papilio example sketches come with schematics showing the chip design inside the BIT file, so you can see what's connected to which pins.

Figure 1. These pulldown resistors ensure that the pin reads low when the button isn't pressed.



things in addition to the ZPU and the output pins: a YM2149, a sigma-delta DAC and an Audio splitter. The first takes the data file and converts it into audio data, the second takes the audio data and creates the signal, and the final one takes a mono audio signal and splits it so that it works with stereo headphones.

The ZPU has 10 empty wishbone slots. These are the places that additional peripherals can connect to the wishbone bus of the processor. In this example, the YM2149 is connected to wishbone slot 5. This is set up in the code with:

```
ymplayer.setup(&ym2149,5);
```

The program then reads data from the SmallFS filesystem. This is included as part of the sketch that's uploaded, and it contains the files in the **smallfs** folder in the sketch folder (these are saved in sketchbook in your home directory). If you have other YMD music files, you can put these in there, and play them, though there's not much space. ZAP does come with one that's loaded by default called **music.ymd**.

Add-on bits of Papilio hardware are known as wings, and these slot directly into the headers on the board. There are a few available; because the FPGA can be reconfigured to include any driver hardware,

Designing chips

In this tutorial we've only looked at using designs that have already been made and compiled for the Papilio. It is possible to create your own and upload these. If you want to understand what's going on at a deep level, there's a free book on the Spartan E3 (the FPGA in the Papilio) and VHDL (a hardware definition language) on GitHub at <https://github.com/hamsternz/IntroToSpartanFPGABook>.

Alternatively, The Gadget Factory has produced a schematics library. This enables you to work at a much higher level and link together components such as the sine wave generator. It is still quite technical, but easier than working with raw VHDL. There are some video tutorials on their website to help you get started: www.gadgetfactory.net/learn.

most of the wings are simple connectors to the hardware (sometimes with a high- or low-pass filter). This sketch is designed to work with an audio wing that includes a low-pass filter and an audio jack. However, this isn't essential. It is possible to connect a speaker directly between the output pins and the ground (though it's best to add a 220Ω resistor to prevent too much current being drawn).

The speaker should be connected to pin 8 on the side with the single strip of connectors (CL 0).

With this in place, upload everything to the Papilio as you did before. First, click on the appropriate BIT file link, and once this has finished, click on the icon to upload the sketch. You should hear the music playing. If it's a bit quiet, you can increase the volume by changing 11 to 15 in the following lines:

```
ym2149.V1.setVolume(15);  
ym2149.V2.setVolume(15);  
ym2149.V3.setVolume(15);
```

You'll need to reupload the sketch for this change to take effect.

You can't arrest me, I'm a rock star

You may have noticed that there are also examples to play Atari MOD files and Commodore SID files. However, these are too big to fit on the Papilio One. If you want to use an FPGA to emulate audio chips, the Papilio Pro is a better choice because it has more space for these more complex designs.

In the next example, we'll keep going with music, but steer away from these more complex designs. Instead, let's get back to basics with a sine wave generator. The one included with ZAP is designed for testing circuits, but it also works for generating tones.

In ZAP, go to Papilio > Papilio Examples > Bency_Waveform_Generator to open the sample project, then go to File > Save As to create a copy.

The software is set up to generate a sine wave at 2.4 MHz, which is a far higher frequency than humans can hear, so before uploading it, you'll need to change this to something a little more audible. Let's go for middle C (or 261.6Hz). Change the line in **setup()** to:

```
setFreq(0.0002616);
```

Burn the bit file to the FPGA, then upload the sketch. You should have a nice clean tone that gets a annoying after a while. To make the tone a little more interesting, we can add a slight vibrato effect by changing the sketch to the following:

```
#define MYBASE IO_SLOT(5)  
#define MYREG(x) REGISTER(MYBASE,x)
```

```
float frequency= 0.0002616;  
float frequencyStep = 0.0000002;  
int numberOfWarbles = 10;
```

```
void setup() {  
  setFreq(frequency);  
}
```

```
void setFreq(float freq)
```

```

{
  unsigned long long phase = freq * 44507433.119;
  MYREG(0) = phase;
}

void loop() {
  for(int i=1;i<numberOfWarbles;i++){
    setFreq(frequency + frequencyStep*i);
    delay(10);
  }
}

```

Most of this is taken from the example sketch, with just a little added to take some values out to variables (to make them easier to change), and a for loop that changes the frequency slightly. This will vary the frequency by gradually increasing it, then dropping it back down. A more sophisticated version would also drop the frequency back down gradually, but we'll leave that up to you to implement.

This produces a slightly more interesting tone, but it still just keeps playing it on and on.

The ZPU has plenty of input/output pins that we can use, and the ZAP environment gives us a way of controlling the sine wave generator based on these inputs. We're going to use this to create a simple keyboard. To save space, we'll only use five keys (four to play notes and one to control the vibrato), but you can use the same techniques to build up to a more complex instrument. The banks of IO pins are handily arranged in blocks of eight, and five of them are completely free (three of the sixth are used up by the wave form generator, but the remaining ones could be used for control).

The code is similar to the example above, with some input pins set up, and some if statement to control the variables depending on which ones are pressed.

```

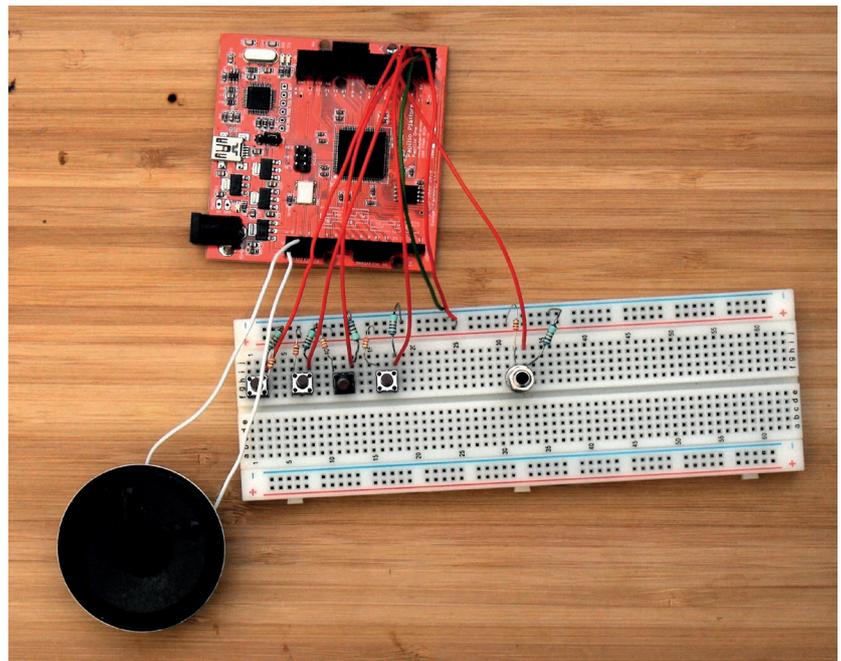
#define MYBASE IO_SLOT(5)
#define MYREG(x) REGISTER(MYBASE,x)

float frequency = 0.0;
float freqC = 0.0002616;
float freqD = 0.0002936;
float freqE = 0.0003296;
float freqF = 0.0003492;
float frequencyStep = 0.0000002;
int numberOfWarbles = 10;

void setup() {
  // put your setup code here, to run once:
  pinMode(0, INPUT);
  pinMode(1, INPUT);
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  setFreq(frequency); //Sets in Mhz
}

void setFreq(float freq) //sets in Mhz
{

```



```

  unsigned long long phase = freq * 44507433.119;
  MYREG(0) = phase;
}

void loop() {
  // put your main code here, to run repeatedly:
  for(int i=1;i<numberOfWarbles;i++){
    setFreq(frequency + frequencyStep*i);
    delay(10);
  }
  frequency = 0.0;
  if(digitalRead(0)==HIGH){ frequency = freqC; }
  if(digitalRead(1)==HIGH){ frequency = freqD; }
  if(digitalRead(2)==HIGH){ frequency = freqE; }
  if(digitalRead(3)==HIGH){ frequency = freqF; }
  frequencyStep = 0.0000002;
  if(digitalRead(4)==HIGH){ frequencyStep = 0.000002;}
}

```

The buttons can't be directly connected to the pins, as they need a pair of resistors in order to correctly control the voltages. See figure 1 for details of how these should be connected. The five input pins on the Papilio are in the middle row at the bottom of the pin bank on the right-hand side if you hold the Papilio with the USB port facing upwards.

We haven't even scratched the surface of what's possible with the Papilio and other FPGAs. The ability to create custom hardware in chips enables you to easily create things that are quite difficult with microcontrollers, and this is only one use for them. At the moment the library of components to go into your designs is still quite small, but it's already exciting. As they become more popular with the hobbyist community, you can expect to see a growing range of easy-to-use schematics. 

Just in case you missed it, hardware wrangler Ben has written a book about Learning Python with the Raspberry Pi. It's called *Learning Python With Raspberry Pi*.

Our little synth will probably never produce any top 10 hits, but it has a sound unique to us.

ERROR DETECTION WITH HAMMING CODES AND CRCs

Data errors are a fact of life, but with a little clever code, you can catch them out before they trip you up.

When sending data across a network or storing it on disk drives, there's a chance that the data will become corrupted. Good hardware design can help minimise this, but it can never eliminate it completely, so it's important to be able to detect when these errors happen so that the data can be re-sent.

The easiest way to check for errors is to add a parity bit. This is an extra bit of data added after each small block that's used to detect errors. There are two types of parity: odd and even. In odd parity, the bit is used to make it so there is an odd number of 1's in the data, while even parity makes it an even number of 1's. When checking the data, the computer just counts the number of 1's and sees if it matches with the type of parity being used.

For example, the letters LV are 01001100 01010110 in ASCII. With an even parity bit added to each letter, they become 010011001 010101100. This adds one bit for every eight bytes of data, so there is more to send; however, in many cases it'll be a good trade off because there'll be fewer errors. This extra data is known as redundancy because it doesn't add any information (we still only have the letters LV,

but we've sent 18 bits rather than 16).

A parity bit won't detect every error. If any single bit gets flipped then it will flag it up, but

if two bits get flipped, it won't detect the error. It will detect an odd number of errors in the data, but not an even number of errors. You could increase the error detection by using a parity bit on a smaller piece of data. For example, you could split the above data into four-bit chunks and add a parity bit on these. This will improve the error detection, but at the expense of extra redundancy. The best trade off between the these two opposing forces will depend on how error-prone the communications channel is.

Error correction

There are other ways than just splitting the data up into ever smaller chunks. One common way is to add three parity bits for every four bits on data. In this case, the parity bits (p1, p2 and p3) are interspersed so that they only cover some of the four data bits (d1, d2, d3 and d4).

- **p1** is calculated based on even parity with data bits d1, d2 and d4.

- **p2** is calculated based on even parity with data bits d1, d3 and d4.

- **p3** is calculated based on even parity with data bits d2, d3 and d4.

The data is sent in the order p1,p2,d1,p3,d2,d3,d4.

By interlacing the parity bits in this way, you can not only tell that an error occurred, but which bit it occurred in. For example, if p1 and p2 are incorrect, but p3 is correct, then you know that the error must be in d2 (or there is more than one error). This means that if you are reasonably confident that the communications channel won't flip more than one bit out of every 7, you can correct any single-bit error. On the other hand, if it's a noisy channel, you can detect any two-bit errors (these would be corrected incorrectly if you dealt with them as single-bit errors).

Stop: Hamming time

This is known as a 7,4 Hamming code, because the method was developed by Richard Hamming, and for every 7 bits set, 4 are data.

The reason we know how many bits can be corrected or detected by a particular code is because of the Hamming distance. This is the number of bits that have to flip to get from one valid sequence to another. For example, using the 7,4 hamming code, 1111 and 0111 are encoded as:

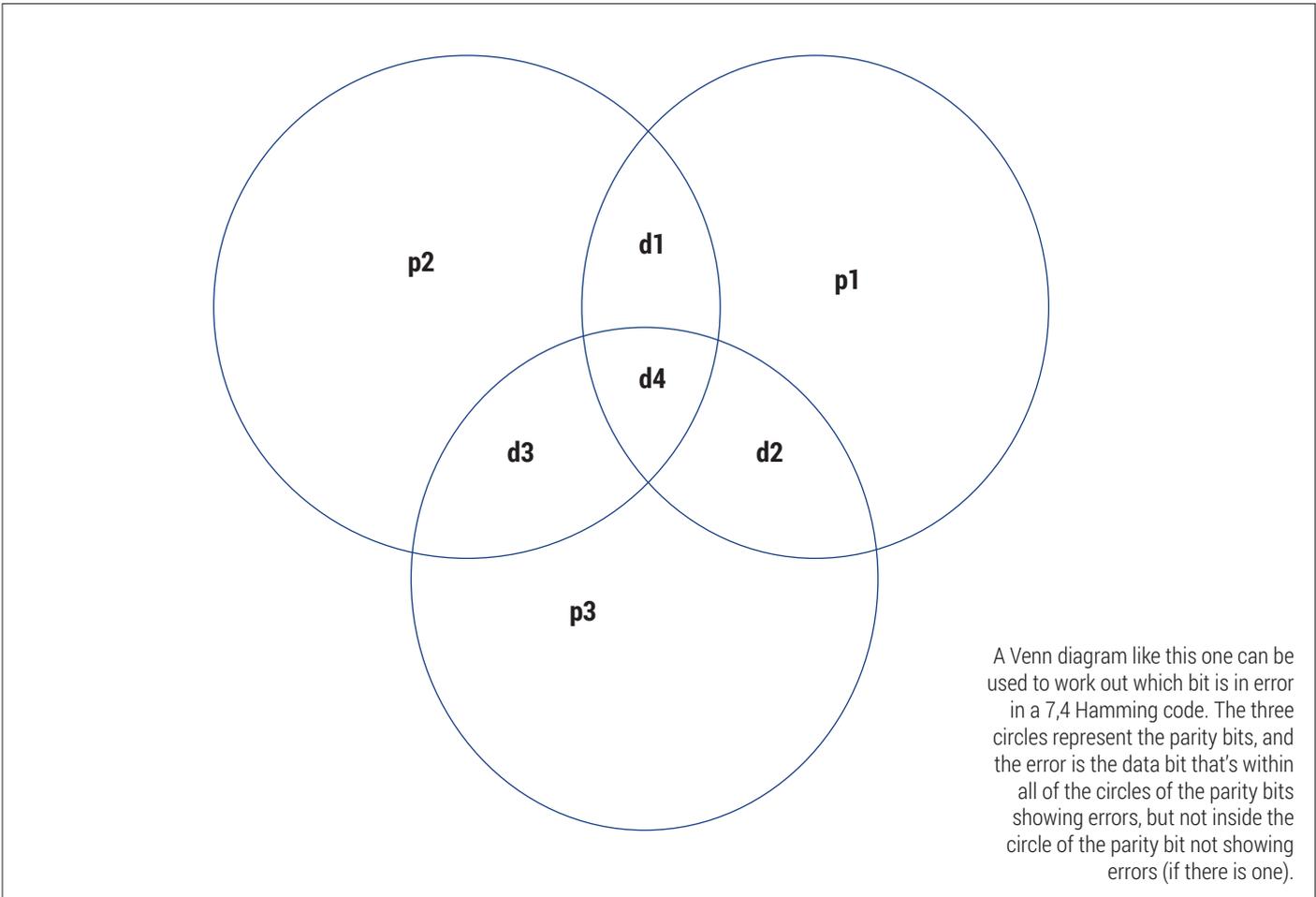
```
1,1,1,1,1,1,1
```

```
0,0,0,1,1,1,1
```

You can see that at least three bits have to flip to get from one to the other. If you calculate 7,4 Hamming codes for all possible values of d1,d2,d3 and d4, you'll find that this is the minimum number of bits different between any two, so we can say that it has a minimum Hamming distance of three. Whenever there is a single-bit error, it will be closer to one code than the other. However, if there are two-bit errors, then it will be closer to the wrong code.

If you want to detect or correct more errors, then you need to increase the Hamming distance. Again, this means increasing the redundancy. It turns out that this is easily done by adding a parity bit that covers every bit (p1–p3 and d1–d4). The result is a code that has four bits of data for every eight bits sent and a minimum Hamming distance of four (this is an 8,4 Hamming code). This can still only correct a single-bit error, but it can detect three bit errors in a

“A parity bit is an extra bit of data added after each small block, which is used to detect errors.”



single block. A 9,4 Hamming code would be needed to correct two-bit errors. It is possible to go on building codes with more and more redundancy that can detect and correct more and more errors. However, the more bits you add, the more you have to send, and so the less data you can get through the channel.

Cyclic codes

Hamming codes are good for detecting errors in very noisy channels, but other times we want a solution that's very quick to calculate and that doesn't have too much redundancy. In this case, we may use Cyclic Redundancy Checks (CRCs). The mathematics of why these work is a little complex, but in practice, they are a little like long division using XOR, and the remainder is the redundancy used to check that the received value is correct. Let's look at an example using the coefficient 1011 on the data 01001100 (the letter L in ASCII) and a three-bit CRC.

```

01001100 000 ← data (trailing 0s are the 3 bits for the CRC)
 1011      ← divisor aligned with the first 1
00010100 000 ← result of XOR
 1011      ← divisor aligned with the first 1
00000010 000 ← result of XOR
 10 11     ← divisor aligned with the first 1
00000000 110 ← result of XOR
    
```

The CRC finishes because the result is all zeros except for the three bits that form the CRC value (110).

To reverse the CRC, the full data is run through the same process, except that the CRC value takes the place of the trailing zeros. At the end, if the remainder is 0 then the CRC data is correct; otherwise there is an error in the data (we'll leave it up to you to perform this and check that the above value is correct).

CRCs don't help you fix the error in the same way that Hamming codes do, they just try to spot them. CRCs can also be designed to be longer or shorter, and the length will make them better or worse at finding errors.

The reason this is very efficient to implement in hardware is that it can be done with little more than a shift register and a few logic gates. This is far less than what is needed for Hamming codes. Because of this simplified requirement, it can run in real time even on very fast data transfer, and you'll find CRCs in everything from CDs and DVDs to Ethernet and 3G mobile networks.

The outside world is a scary place for data. There is all manner of electromagnetic interference and other problems just waiting to flip 0s into 1s. With a little careful planning, though, we can protect our precious digits and make sure they don't come to any harm. 

Ben Everard is the best-selling co-author of the best-selling *Learning Python With Raspberry Pi*.

DEVELOP YOUR FIRST ANDROID APPLICATION

Google's Android Studio has made Android development more straightforward than ever. As we prove. Kind of..

WHY DO THIS?

- Develop for a hugely successful platform
- Get into the Android app store and make £\$€
- Never miss another Linux Voice podcast

Android's success is colossal. It's the most widely distributed version of a Linux-based operating system, used on everything from smartphones and tablets to cars, fridges and air conditioning units. It's only a matter of time before it reaches your toaster. And while it may have started life outside of Google as an advanced operating system for digital cameras, Google's purchase of Android, Inc. in 2005 slotted Android into Google's plans for mobile world domination, inadvertently making it the operating system for the touchscreen generation – the opposite of Apple's iOS, with all its walled garden and hardware consistency. Android is wild, chaotic, uncountable and outgrowing itself. But it's also open source and still relatively open. You don't need to pay Google to write software for it, and you don't need to root your device to run your applications. Despite an unwieldy API and a huge boilerplate of

requirements for running your own applications, development isn't even that difficult – especially if you just want to hack your own solutions into the chaos. Which is exactly what we're going to do here, in what will be the first part of a series on creating a Linux Voice application for Android.

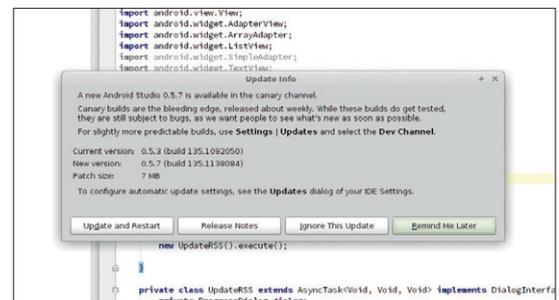
We're going to start modestly. We'll build a very simple RSS reader app that will parse the latest posts from **LinuxVoice.com** and enable you to select a story to open in a browser. As a starting point, we may then expand upon this simple application to create a much more functional application. But it's also a perfect way to acclimatise yourself to Android development, and from there, put your own ideas into code. And because we're embracing all things cutting-edge and chaotic in this tutorial, we're going to use this as an excuse to try Google's own Android development environment – Android Studio.

1 INSTALLATION

Android Studio exists solely for Android applications. It ties itself seamlessly with the Android SDK and the emulator tools used to test your code, and also offers cutting-edge shortcuts for writing. There's WYSIWYG realtime app rendering, a UI designer, Android templates and an editor that spots mistakes as they happen. But its cutting edge nature also means it's unstable, and Google doesn't yet recommend it for use in mission-critical development. That's not going to affect us as we take our preliminary steps in Android development, but it's something to consider when your app development ambitions take you further than a simple RSS reader.

Android Studio is based upon another IDE, IntelliJ IDEA, and as you might have inferred by the mention of Eclipse and IntelliJ IDEA – two IDEs written in Java, Android is all about the Java. Java is something of a contentious language even now, but there's no denying its ubiquity. More importantly, as long as you've some previous experience in an object-oriented environment, Java is legible and easy to understand and get along with. It's a good choice for a hacker-friendly tutorial.

We installed Android Studio on both Arch Linux and Linux Mint/Ubuntu. Arch packages are held in the user repository, while manual installation is also relatively straightforward: just download and execute a binary.



It's worth updating Android Studio frequently, as each update makes the environment more stable.

However, we find installation on Mint an easier proposition as it automatically deals with dependencies and the environmental variables required by Android Studio to find the SDK (which also needs to be installed manually). To install from Ubuntu or Mint, just type the following:

```
sudo apt-add-repository ppa:paolorotolo/android-studio
sudo apt-get update
sudo apt-get install android-studio
```

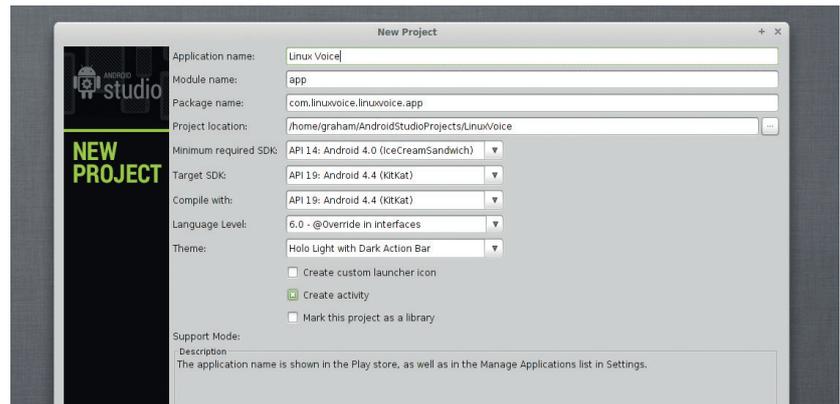
This should grab over 650MB of data, as the download includes the SDK alongside the development environment and any dependencies you've yet to install. But it will leave you with a full development environment capable of building

projects that can run on your desktop through an emulator and deployed to your own Android devices. Android Studio can be started by typing **android-studio** on the command line, or through an icon embedded within your desktop's launch menu, and the first thing we'd recommend you do after the splash screen has gone is to ignore the 'Welcome to Android Studio' window, and instead open the small 'Check For Updates' link in the windows bottom

border. After checking against the server, the application will almost certainly announce that there's an update that can be installed, and unlike other applications installed through a distribution, in-place updates to Android Studio work. Just let the application run and update itself. Android Studio is in a rapid state of development, so it's a good idea to keep on top of updates. There are usually one or two per month, and recent updates have been very stable.

2 DEVICE CONSIDERATIONS

From the 'Welcome to Android Studio' window, click on 'New Project' to start our adventure. We've now got to change a few values in the next window that appears. The app needs a name, and we called ours simply 'Linux Voice'. We'll also need a package name, which needs to adhere to Java's package naming convention. Recent versions of Android Studio will automatically create this from your application name, but if you need to change this to something that closer matches your own configuration, make sure it uses the reverse hierarchical naming pattern, normally (but not always) linked to the top level domain of your organisation plus the name of your app.



The next three fields ask you which version of the SDK you want to build your application against. Android has changed a lot over the years, and many of its more advanced features are only available in the 4.x SDK. But while choosing the latest SDK is the easiest option, the best answer is going to depend on what capabilities you want your application to have and where you want it to be used. Unlike iOS, many Android users can't simply update their devices to the latest releases. Many manufacturers don't even provide updates, and even when they do offer updates, they've usually made so many modifications to Android themselves that it takes many months for an update to be distributed. The result is that there are many different versions of Android in active use.

using a 4.x version of Android, there's still almost a fifth of all Android devices running Gingerbread. We decided with setting a minimum required SDK of 14 (Ice Cream Sandwich), while setting the target and compile SDK at the very latest version, 19 (Kit Kat).

The Android Studio New Project wizard creates sensible defaults for nearly all values.

Choose your platform

Google uses statistics gathered from its Google Play app to report on the devices accessing the store over the previous seven days, so that developers can make their own judgment on which platforms to target. We've printed the table from early April, and you can see that while a significant percentage of devices are

You can leave the other options at their default values – just make sure that 'Create Activity' is enabled, as this adds the next page to the New Project wizard, which enables you to select which kind of template to use for your project. These are worth exploring, as each example will include the best practice for different elements of a typical Android app, and each new version of Android Studio includes more. But for our first application, we're going to keep things simple and go with the 'Blank Activity'. This lets us build things up from scratch but also avoids too many complex concepts that come with more complex templates. After selecting the blank template, the final page is where we give our activity a name.

Current Android versions			
Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%

An activity is what Android calls the application component that expects something of the user. For nearly every instance, that means putting something on the screen you can interact with. Android likes to break down the functional components within an application, and an activity is one such component, usually with one purpose. It could be a web browser, or a music player, but equally, it could be a stream of RSS stories, which is the only activity our application is going to provide. An application is then a combination of multiple activities tied together to create a fully functional tool. Having said that, you can leave the Activity and Layout names at their default values, as they make only a cosmetic difference.

LV PRO TIP
When using 'try' and 'catch', debugging is much easier if you isolate each exception rather than grouping them together (as we have).

3 ANDROID EMULATION

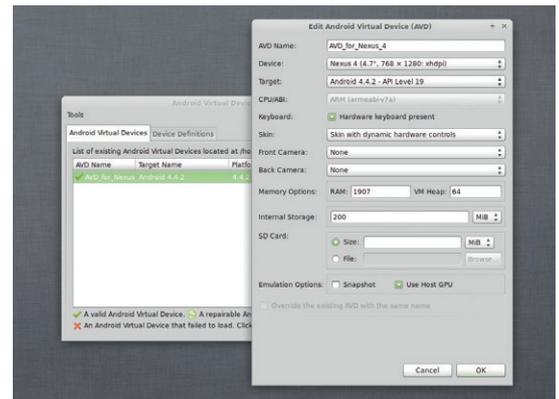
This is where we hit some cutting-edge turbulence. With Android Studio 0.5.7, the template generated an error before we'd even got to the code. The error console complains that the Android Support Repository isn't installed. And while this was indeed installed, any update required another update to push the Support Repository version up from 4 to 5. To fix this, we needed to open one of several support tools used to augment the Android development environment – the Android SDK Manager. This can be opened from the Tools > Android menu, and when opened it lists the various SDKs and tools installed alongside Android Studio. We needed to allow this to update two packages, which it did automatically after accepting a couple of licences from Google. Hopefully, you won't have to go through a similar process when you first start Android Studio.

Before we spend a little time going through Studio's GUI components, first click on the Play button in the toolbar at the top of the window. This will build and attempt to launch the template, which is itself a fully executable application. Most of the few seconds it takes is spent launching the Android Debug Bridge, ADB, which is a tool that grabs runtime and debugging information from a runtime instance of your application, whether it's running on an emulator or on a real device.

PRO TIP
 Android Studio's editor will grey-out lines, such as imports, if they're redundant, and even make suggestions for spelling corrections.

Android emulator

The next window that appears lets you select which device you want to run your application on. The Android Emulator is one of the most useful tools for both Eclipse and Android Studio development, as it enables you to run your code on a fully fledged Android device, albeit one being emulated in software. This does affect performance, but it also means you can spin up any kind of device you want, from old models using an ancient version of the SDK, to models with resolutions and hardware combinations that don't exist, and you don't have to spend any money getting the latest devices. From the 'Choose Device' window, make sure 'Launch Emulator' is



You can run almost any real and functional Android device from the Android Virtual Device manager.

selected and click on the 'Expand' button to the right of the drop-down list. This will open AVD – the Android Virtual Device manager.

This is the tool that manages your real and fictitious virtual Android devices, and to get started you need to click on the 'Device Definitions' tab. This page lists common definitions for a device, and clicking on 'AVD' will create an instance for you to run. If you want to edit any of these settings, you'll need to clone a definition first. This will allow you to adjust things such as the native resolution of a device, or whether both portrait and landscape modes are available. After you've added a device to create an AVD, you can edit runtime options such as the amount of internal story available to the emulation, whether an instance is persistent (storing the data between sessions), or whether graphics calls use your real machine's GPU, as well as the target level of the SDK.

If you've got the correct packages installed, you can also switch between the much slower ARM CPU emulation to the x86 instruction set of your native system. This gives a big performance boost and won't cause any compatibility issues unless you're performing more complex or low-level operations. We opted for a clone of the Nexus 4 profile with no other modifications.

4 TESTING YOUR ENVIRONMENT

With a device added to the device list, you just need to click on Launch to run the instance. However... stop! On most displays, a running Android instance takes up nearly all your screen space, because of the high-DPI values on most Android devices. A Nexus 5, for example, has a screen resolution of 1920x1080 pixels, which is more pixels that many screens are capable of displaying. We've not found a way of scaling the emulator from the launcher, but you can scale the emulator from the command line. For that reason, we'd recommend running it from there. If the

Android tools have been added to your path, it can be run with the **emulator** command; if not, you'll need to find it yourself. Arch installs the SDK into the **/opt** folder, while the PPA packages for Ubuntu and Mint install themselves into **/usr/share/android-studio/data/sdk/**. And because your own copies of the AVD profiles are copied to your home folder, you can reference them directly from the **emulator** command. To run the emulator at 50% resolution, which we've found perfectly acceptable, type the following, replacing the name of the virtual device with the AVD

name listed in the AVD manager:

emulator -scale 0.5 @AVD_for_Nexus_4

The emulator window will appear at the proper scale and proceed to load Android. This can take a minute or two if you're emulating an ARM CPU, but speed obviously depends on your system. Before too long, you should have a working Android system that you can unlock by sliding the padlock to the right with a click of the mouse.

You should also be able to go back to the 'Choose Device' window from Android Studio and see the running instance listed beneath the 'Choose A Running Device' button, which should be enabled. To run your application, select the running device from the list and click on OK. We'd also suggest you

enable the 'Use Same Device For Future Launches' so that you don't have to go through this step again - although this only works for the current session.

A few moments later, your virtual Android device should burst into life and your application will run automatically. This is the default testing environment, and if you take a look back in Android Studio, you should see that it's in constant contact with the emulator. When you start debugging or monitoring your application, you'll be able to break execution and watch variables across the Android Debug Bridge exactly as you could if the application was running natively on your desktop. If everything is working correctly, you should see the ubiquitous "Hello world" staring back at you, and it's now time to start coding.

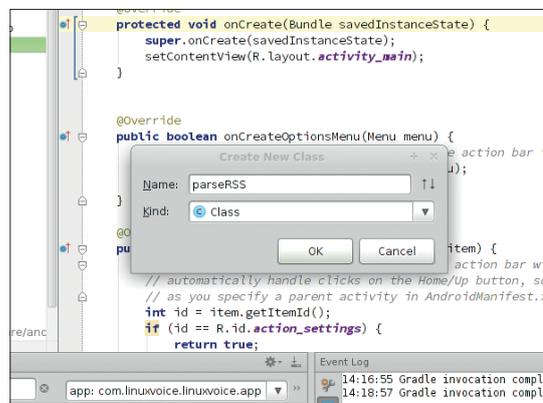
5 LET'S START CODING!

Before we start with the code specifics, we need to allow our application to access the internet (on the device you eventually run it on). Think of this being the Android equivalent to SELinux, and all such control is governed by a single file found in the root directory of your project – **AndroidManifest.xml**. This file controls access and permissions as well as what can and can't be received by the various activities that make up your application. You can edit it in Android Studio from the project hierarchy view on the left, by clicking on the project name, 'app' then 'src'. You need to add the following before the '<application' tag, as this ensures accesses to the entire package:

<uses-permission android:name="android.permission.INTERNET" />

With that line edited, we can now start with our own code. The first thing we want is a method to grab and parse the RSS provided by our site, and as this is such a common requirement, many people have already solved this problem in lots of different ways. The most straightforward we found was written by Tony Owen (<http://droidapp.co.uk>) and used in several open source projects, including the now rather old rpdroid podcasting client. Tony offloads RSS handling into its own Java class, and we'll do a similar thing. To create a new class, follow the path hierarchy in Android Studio down through src, main, Java and your package name. You should already have a single activity called 'MainActivity', unless you changed this in the startup wizard. MainActivity is where all the action is happening, and where you can quickly augment the "Hello World" app with your own features.

To create a new class (which will be contained within its own 'Java' file), right click on MainActivity and select 'New > Java Class'. In the small window that appears, enter a new class name – we used 'dataRSS', but as long as you're consistent within the application, it won't matter which name you choose. You'll then see the new class listed beneath MainActivity, with the text editor waiting for your



Create a new class within Studio by right-clicking on the main activity source file.

inspiration. This class is going to be very simple, with no methods to do calculations and only four variables – three to hold the string parts of the RSS feed we want to pass back to MainActivity, and a single integer to keep the list location of the eventual post when it appears in our application. The only peculiarity, if you're not used to Java, is that our strings are arrays that we don't have to give a size to until we initialise them later:

```
public class dataRSS {
    public static String[] postTitle;
    public static String[] postURL;
    public static String[] postContent;
    public static int position;
}
```

We're going to follow this with the only other class we're going to write, a class we've called parseRSS, which should be created the same way we created dataRSS. This will contain the method that will download and parse the bits we want from the RSS file, so it's going to be slightly more complex. Let's go through the code, starting with the first few lines:

```
public class parseRSS {
    public static void parse() {
        URL url;
```

```
try {
```

All we're doing is creating the class and adding a single method, which is the function we're going to call to process the RSS. You should notice that as you add 'URL', it gets highlighted in red. This is because the Java compiler can't find any other reference. Hold the cursor over the URL characters and the editor will complain it can't resolve the symbol. The editor in Android Studio is smart and can make lots of intelligent calls when spotting and fixing errors. If you now click the 'URL' text, the tooltip should say 'Java.

net.URL? Alt + Enter'. The editor is saying that Java.net.URL satisfies the resource and can be imported by pressing Alt + Enter. If you do this, a popup menu will open and the first option will allow you to import the class, and selecting that will add 'import java.net.URL;' to the top of the file and solve the error. For this reason, we haven't included any of the 'import' statements for our code, as you should be able to add them yourself (and it saves space). If you have problems working them out, download the project source code from LinuxVoice.com.

6 MAKING A CONNECTION

The last line in the previous snippet is **try**, and it's one of Java's most useful functions. It's an exception handler that forces us to deal with the consequences of something not happening rather than allowing the app to crash. The situations dealt with by **try** will be dealt with by a **catch** instruction later on in the code.

In the next two lines, we'll give **url** the address of our feed and ask Java to open a connection:

```
url = new URL("http://www.linuxvoice.com/feed");
URLConnection conn = (URLConnection) url.
openConnection();
```

If the connection can be made successfully, our application will get past the following 'if' statement, taking us into the section of code that deals with the nitty gritty of the XML within the RSS feed. There are many ways to deal with RSS/XML, but we've found the easiest to be DocumentBuilder, which provides an API to access the tree layout of an XML file. In the following code you'll see that we set up a few variables before structures for getting to the data within the data stream before **db.parse(url.openStream())** passes control of the data to the variable we call **doc**. We'll also initialise the arrays we created in the **dataRSS** class using the size of the list gleaned from the RSS and now held in **itemLst**:

```
if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.
newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document doc;
    doc = db.parse(url.openStream());
    doc.getDocumentElement().normalize();
    NodeList itemLst = doc.getElementsByTagName("item");
    dataRSS.postTitle = new String[itemLst.getLength()];
```

```
dataRSS.postURL = new String[itemLst.getLength()];
dataRSS.postContent = new String[itemLst.getLength()];
```

We're now going to go through **itemLst** using a 'for' loop and attempt to assign the contents of the elements referred to by the tag names 'title', 'link' and 'content:encoded' by passing them to **NodeList** elements, which is simply a collection of nodes we hope will contain the data we're after.

```
for (int i = 0; i < itemLst.getLength(); i++) {
    Node item = itemLst.item(i);
    if (item.getNodeType() == Node.ELEMENT_NODE) {
        Element ielem = (Element) item;
        NodeList title = ielem.getElementsByTagName("title");
        NodeList link = ielem.getElementsByTagName("link");
        NodeList content = ielem.getElementsByTagName("content:en
coded");
```

Because we can't be certain that the data we're after will be contained within the title, link and content **NodeLists** we've created, we'll need some exception handling again with **try** as we attempt to assign the values from the RSS to the string arrays we created:

```
try {
    dataRSS.postTitle[i] = title.item(0).getChildNodes().item(0).
getNodeValue();
    dataRSS.postURL[i] = link.item(0).getChildNodes().item(0).
getNodeValue();
    dataRSS.postContent[i] = content.item(0).getChildNodes().
item(0).getNodeValue();
} catch (NullPointerException e) {
    e.printStackTrace();
}
```

In the previous piece of code, you'll see that we've included the **catch** block for dealing with the exception. As you might expect, all this does is output the methods (the stack trace) of the application that have led to this point, and we'll finish off the class with closure on the previous **try** as well as close brackets for all the other blocks. We've put these on the same lines for brevity, as you should be properly indenting your code or getting Android Studio to do it for you (Code > Reformat Code in the menu):

```
}} }
} catch (Exception e) {
    e.printStackTrace();
}} }
```

All our import lines were automatically added via Android Studio's auto-complete features.

```
package com.linuxvoice.linuxvoice.app;
import java.net.HttpURLConnection;
import java.net.URL;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

7 ADDING THE FUNCTIONALITY

We're not going to spend much time on the GUI this month. All we're going to do is replace the TextView widget with the ListView widget. ListViews are now synonymous with smartphones – they're the bread and butter widget used to display everything from email to Tweets. And while Android Studio has a wonderful visual editor for creating your own user interfaces, the best way to accomplish this is to open the GUI resource in your Android project – **app/src/res/activity_main.xml** in the project's hierarchy. When you click on this file, Android Studio will switch to the visual editor, but you'll need to switch from the 'Design' tab to the 'Text' tab to edit the XML file responsible for the layout directly. This view is still excellent, as it updates a virtual Android device of your choice, giving you the best of both worlds. To replace the TextView, simply paste over the entire block with the following:

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView"
    android:layout_gravity="center_vertical"
    android:layout_weight="1" />
```

The important thing to note here is that we're calling the new widget **listView**, as referenced by the **android:id** tag. We'll use this to reference the widget within our MainActivity code, and that's going to be our next and final target for this tutorial, so double click on **MainActivity.java** and let's make it so!

Before we get onto the complete code for the main class, we've removed both of the methods that deal with the menu, because while they're an essential part of the template for adding functionality, they're only going to add complexity here.

```
public class MainActivity extends Activity {
    parseRSS parserss;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        parserss = new parseRSS();
        new updateRSS().execute(); }
}
```

The above section of code is the beginning of our class. We create our own global **parserss** variable from the parseRSS method we created earlier – this should be important automatically because the Java class is in the same file as MainActivity. We then change the contents of the **onCreate** method that Android Studio creates automatically, adding the **parserss** activation and an **execute()** method call for a background processing class we haven't created yet. The **@Override** annotation is already there, because this is a reimplementing of another method and we want to make sure Java executes our version instead.

The next stage is to write the method to add items to the ListView widget, and to handle what happens when one of those items is clicked to add items to the

list. This is quite straightforward. We use **findViewById** to link the UI widget we laid out earlier with the ListView widget we create in code, and call this 'storylist'. We're then able to add elements via the **setAdapter** method. We're just adding the title at the moment, but we use the URL in the next view lines. We need to create an even listener to deal with user input, and we use ListView's **setOnItemClickListener** and the **AdapterView.OnItemClickListener()** to handle the callback, which then decodes which item has been pressed via our 'position' variable and starts a new activity based on the URL. If we linked to an audio file, Android would launch an audio player, but as we're linking to a web page, it's going to open the web browser. You'll always be able to return to our app using the 'back' button.

```
public void populate_list() {
    ListView storylist;
    storylist = (ListView) findViewById(R.id.listView);
    storylist.setAdapter(new ArrayAdapter<String>(MainActivity.this,
        android.R.layout.simple_list_item_1, dataRSS.postTitle));
    storylist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
            dataRSS.position = position;
            Uri uri = Uri.parse(dataRSS.postURL[dataRSS.position]);
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            startActivity(intent);
        } });
}
```

The final chunk of code is the updateRSS method that we called **execute()** on earlier. It uses AsyncTask to show an update dialogue and stop our application from locking up whilst we update the RSS feed.

```
private class updateRSS extends AsyncTask<Void, Void, Void>
    implements DialogInterface.OnCancelListener {
    private ProgressDialog dialog;
    protected void onPreExecute() {
        dialog = ProgressDialog.show(MainActivity.this, "", "Updating RSS. Please wait...", true); }
    protected Void doInBackground(Void... unused) {
        parserss.parse();
        return null; }
    protected void onPostExecute(Void unused) {
        dialog.dismiss();
        populate_list(); }
    public void onCancel(DialogInterface dialog) {
        cancel(true);
        dialog.dismiss();
    } }
}
```

And that's it! Run the project and it should download our RSS feed. Click on a story and it will open it from your default browser. Let us know how you get on! 

Graham Morrison is the author of Kalbum, a photo collection manager that, in its heyday, was in the Mandriva repositories. Nowadays he's best known for synthesizer music.

LINUX VOICE MASTERCLASS

Essential Linux tools explained – this month, the VLC media player and a trio of command line media tools

VLC MEDIA PLAYER – THE ONE THAT DOES IT ALL

Play media, transcode files, stream live media and more with this ancient application.

JOHN LANE

The VLC Media Player is a free and open-source multimedia player. There are a number of Linux media players to choose from but this one can play just about anything and will often succeed where other players fail.

But it is much more than just a media player. It was originally a pair of applications: the VideoLAN Client called **vlc** and the VideoLAN Server called **vlc**. The client and server functionality are now both contained in a single application – the VLC Media Player, which everyone now refers to as VLC.

You can use VLC to encode videos in a similar way to the command-line tools that we looked at earlier and you can use it as a streaming server. But you'll probably want to start by using it as a player.

Playlist

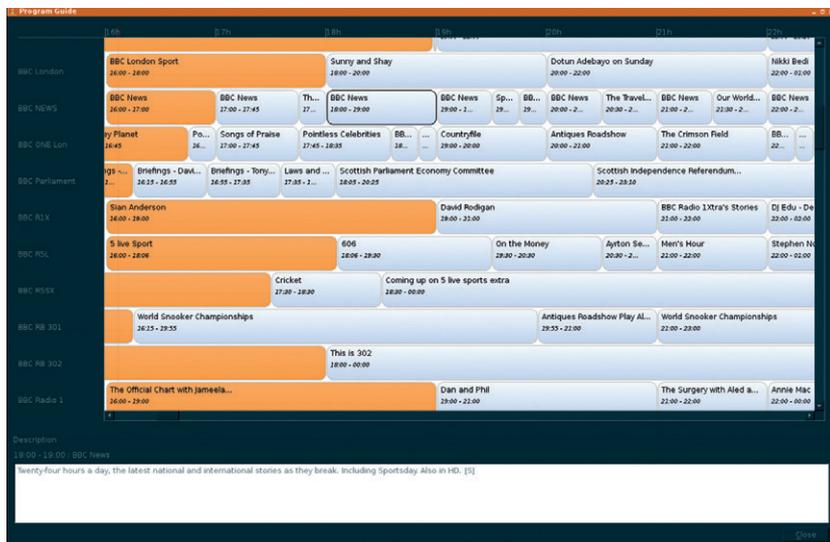
Central to VLC is its playlist, which is more than your typical playlist – you can create, load and save your own playlists, but what we have here is more of a gateway to content, both on your local network and

on the internet. You can directly access media on your local machine, either its hard drive or DVD. If you plug in a media player that uses the Media Transfer Protocol (MTP) you can access its contents directly.

If you have any DLNA (Digital Living Network Alliance – we know, it sounds horrible) devices such as a PVR machine on your network, then you can also access their content through the playlist. DLNA is a media sharing standard offered on some internet-ready audio-visual devices like televisions and PVRs, and it is a simplified subset of something called Universal Plug And Play, which VLC also supports. Opening the playlist (Ctrl+P) and selecting Universal Plug'n'Play will query the devices on the network (this may take some time) and then offer a list of content. Click on an item to play it. Other network services can be accessed in a similar way – VLC supports SAP, the Session Announcement Protocol, which gives access to published multicast services and Apple's Bonjour protocol.

You can even directly access Freeview digital television broadcasts if you have a working TV card and know your local transmitter's frequencies. Go to Media > Open Capture Device and select the TV – Digital capture mode. Enter the relevant frequency and press Play. Programme guide information is collected and you can view this with Tools > Program Guide.

VLC's programme guide lets you see what's on TV.



Internet media

Beyond your local network and TV antenna, the internet is a rich source of media, and VLC gives easy access to internet radio sources like Icecast and Jamendo through the Internet choices on the Playlist screen. You can also add your own media favourites, such as your favourite Linux podcasts.

To add a podcast, go to View > Playlist, then click on Podcasts; the Plus sign opens a dialog where you can enter the URL for the podcast (for example, www.linuxvoice.com/podcast_opus.rss), giving you

direct access to all episodes of the podcast that the feed provides.

The default user interface fits in with your desktop environment, but VLC is skinnable, meaning that you can change its visual appearance. Many skins are offered at the VLC website, www.videolan.org/vlc/skins.php, and you can even download them all at once like so:

```
mkdir -p ~/.local/share/vlc/skins2
curl http://www1.videolan.org/vlc/skins2/vlc-skins.zip | bsdtar
-C ~/.local/share/vlc/skins2 -xvf-
```

You need to go to the Preferences page (Control+P) within VLC and set the Look And Feel to “Use Custom Skin” and then exit and re-start. You can then choose a skin by right-clicking Interface > Select Skin.

You can also choose a skin when starting VLC using the `--skins2-last` command-line option. You need the full path to the skin file:

```
vlc --skins2-last=~/.local/share/vlc/skins2/neon.vlt
```

Be aware, however, that using skins can hide some of VLC’s functions.

Broadcast yourself

Once you’re happy that you can play pretty much anything you want, you can move on to streaming, because VLC will stream anything that it can play. You can select anything from your playlist and stream it – just use the drop-down menu on the Play button to change it to Stream.

You’ll be walked through a set of screens where you specify what and how to stream – the protocol to use and any transcoding to perform. The first screen defines what will be streamed, and you can fill this in yourself, but selecting from the playlist will fill it in for you. The second screen enables you to define streaming destinations. These can be a file or one of several network profiles including HTTP and RTP (you can have more than one if you need to). You can also choose whether the streamed content is displayed locally so you can see what’s being streamed.

The fourth screen is where you can enable transcoding, and there are several configurations to choose from. If none of those suit, you can create your own. Once the stream starts, you can view it from any player that can access it (another copy of VLC is an obvious choice here).

Streaming to a file enables you to record a live stream to your local machine. You can also save remote content locally, which is a good way to take a copy of recordings from your PVR onto your computer. You can transcode while doing this if you want the copy in a different format to the original.

Your preference

We already mentioned how you can change the look and feel with custom skins. You can also change the behaviour with Preferences (Control+P). There are many options available here to change VLC’s behaviour, but it’s unlikely that you will need to change many of them. If you want to back up your



preferences, they’re stored in a file:

```
~/.config/vlc/vlcrc
```

One of the preferences controls the video output. It’s normally set to automatic but, for fun, try setting it to Color ASCII art video output. You need to re-start VLC after changing preferences but, once you’ve done this, any video played will be rendered using colour ASCII art. This isn’t much use unless you’ve only got a text terminal, but it’s fun nonetheless and just goes to show that VLC has many tricks up its sleeve.

You can use the command-line version of VLC to view a movie in any terminal window. Now you don’t even need a graphical desktop to watch things! (press Alt+F4 to quit.)

```
cvlc -V caca my_movie.mp4
```

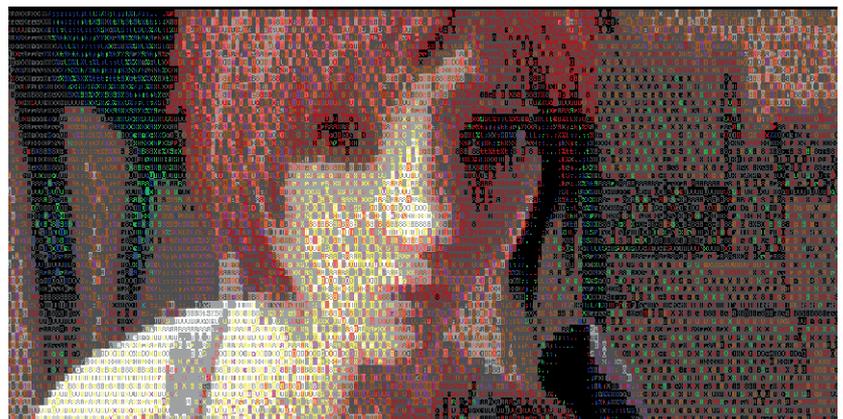
No GUI needed

If you’re using VLC for downloading or streaming or you have another reason why you don’t need the full GUI experience then

there is a command-line version called CVLC, which offers the same functionality but is driven through command-line options. This is its server heritage showing through, and CVLC enables you to use VLC for non-GUI applications such as setting up a streaming server or as a transcoding tool.

“Streaming to a file with VLC enables you to record a live stream to your local machine.”

With VLC’s ASCII art, you can watch a movie on a terminal.



FFMPEG AND MENCODER: COMMAND LINE VIDEO

Use your command-line prowess to make media files for any device.

As with most things Linux, there is a perplexing number of media tools available. Many of these are merely wrappers around a small number of utilities that perform the heavy lifting and the most common ones include Mencoder and FFmpeg, which we're going to look at here.

Mencoder comes from the same stable as the MPlayer media player, and it enables audio and video files to be decoded, filtered and re-encoded. As it's built from the same codebase as MPlayer, if MPlayer can play it, Mencoder can decode it.

FFmpeg is a collection of tools and libraries for encoding, decoding and streaming audio and video. It includes the libavcodec library, which is widely used by other packages including Mencoder. FFmpeg claims to be able to work with pretty much any multimedia file.

In addition to **ffmpeg**, the command-line tool for converting multimedia formats, the package includes a streaming server called **ffserver** and a basic player called **ffplay**. Use **ffprobe** to display media information about a file.

Both Mencoder and FFmpeg are complex and you will often find that using internet searching will offer you the command line sequences that you need to achieve a task and those search results will determine the tool you use.

Both tools are incredibly versatile. They are driven by a large number of command line arguments and this complexity can, at first, present a barrier to new

“Ultimately, no GUI app can be as flexible as a comprehensive command line tool.”

users. We'll introduce some of the more common arguments and refer you to the man pages and web-based documentation where you can learn more.

The natural complexity of these tools has brought about several GUI-based tools that try to simplify their use, with varying degrees of success. Ultimately, no GUI application can be as flexible as a comprehensive command line, and these tools present a good case for taking the time to learn their complexities instead of masking them with a GUI.

Codec container

These tools manipulate multimedia files – containers housing one or more streams of encoded data, usually audio and video and sometimes data such as subtitles. Encoding stores audio or video as a bitstream that can later be accessed by a decoder.

The software that performs encoding and decoding is commonly referred to as a codec, and is defined by standards that can have multiple implementations. These implementations all produce acceptable input for any decoder, but may differ in the way they do it.

One popular codec these days is called H.264, otherwise known as MPEG-4 Part 10 or AVC, the Advanced Video Codec. The usual open-source implementation is called x264. H.264 is very good at producing high-quality video with small file sizes and is therefore the current codec of choice for the internet and mobile devices. Blu-ray discs are also encoded using H.264.

The file itself is a container, and there are various container formats that you will encounter including AVI, MP4, MKV and Ogg. You will usually find H.264 within an MP4, MKV or MOV container.

Both Mencoder and FFmpeg use the same libraries for the codecs (**libavcodec**) and containers (**libavformat**).

Enter the command line

So, how do we use them? Basic usage requires that you supply an input file, some options and an output file, like this example:

```
mencoder input-file -oac mp3lame -ovc lavc -o output-file.avi
```

The **mencoder** command reads the **input-file** and uses the given audio and video codecs to re-encode its contents, writing to the specified **output-file**. An **ffmpeg** command line sample to achieve a similar thing would be

```
ffmpeg -i input-file -acodec mp3lame -vcodec lavc -o output-file.avi
```

The **input** can be any file. The formats contained within the file are detected automatically. **ffmpeg** will display the format information if run without output instructions:

```
ffmpeg -i input-file
```

Alternatively, the supplied **ffprobe** command does the same thing:

```
ffprobe input-file
```

As an example, if you can't play our free-and-open Ogg Video files on your smart television then don't fret – just convert them into a MPEG transport stream that your TV will understand. Either

```
mencoder fosspick3_cable3.ogv -oac lavc -ovc lavc -lavcopts vcodec=mpeg2video:acodec=mp2 -of lavf -lavfopts format=mpegts -o fosspick3_cable3.mpg
```

or

```
ffmpeg -i fosspick3_cable3.ogv -vcodec mpeg2video -acodec mp2 -f mpegts fosspick3_cable3.mpg
```

will give you an MPEG2 Transport Stream container that should be playable on your TV, because it uses the same mpeg2video and mp2 audio codecs that broadcast digital TV uses.

Choose your command

These examples show the style of the command-line options for each tool and highlights the subtle differences between them. The main difference is that Mencoder selects the codec library (**-oac** and **-ovc**) and then passes options in to them (**-lavcopts**). Because **ffmpeg** uses the library directly, its arguments are more clearly presented as **ffmpeg** options (**-acodec**, **-vcodec**, **-f**).

You can query Mencoder and FFmpeg to see their supported codecs and container formats:

```
mencoder -oac help -ovc help -of help
```

```
ffmpeg -formats
```

```
ffmpeg -codecs
```

The command line arguments **-oac** and **-ovc** take parameters that select the audio and video codecs to use, but here we instead use **help** to see what's available. The third argument, **-of**, selects the output container format, and the most useful of these is **lavf**, which enables you to use any container format that is supported by FFmpeg's **libavformat** library. Normally the format is inferred from the output filename, but it can be stated explicitly, as we did in our command line examples above.

FFmpeg can display more detailed information about a codec, for example

```
ffmpeg -h encoder=mpeg2video
```

lists all the command line options applicable to that codec. You will see that there are many options available.

You don't have to use an input file. Using the **x11grab** option enables you to capture a screencast directly from your desktop:

```
ffmpeg -f x11grab -r 25 -s 1024x768 -i $DISPLAY -vcodec huffyuv screencast.avi
```

Here we explicitly set the input format and set the source to our X display. The **huffyuv** codec is lossless and captures the screen exactly as-is. We also set a frame rate and screen size. To make the resulting video useful, we should transcode it into a more useful format such as H.264.

```
ffmpeg -i screencast.avi -vcodec libx264 screencast.mp4
```

This time, the input format is detected automatically from the input file. We just need to specify the video codec and output container.

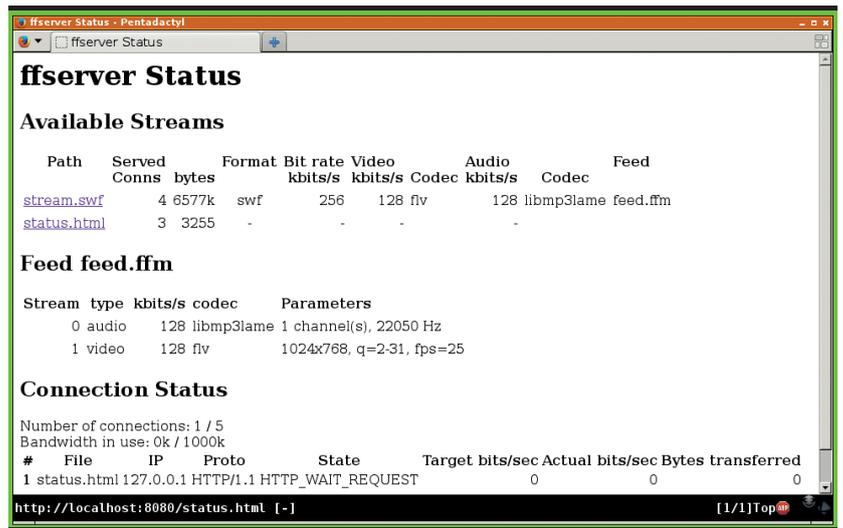
Live video streaming

As well as encoding, FFmpeg can also stream content. Its **ffserver** component accepts feeds from **ffmpeg** clients and streams them over HTTP. Setting up a server requires a simple configuration file called **/etc/ffserver.conf**:

```
Port 80
```

```
<Feed feed.ffm>
```

```
File /tmp/feed.ffm
```



ffserver is FFmpeg's streaming server. You can view its status in a browser.

```
</Feed>
<Stream stream.swf>
  Feed feed.ffm
  Format swf
  VideoCodec flv
  VideoBufferSize 80000
  VideoSize 1024x768
  Noaudio
</Stream>
<Stream status.html>
  Format status
</Stream>
```

This minimal example sets up an incoming feed and an outgoing Flash video stream. It also sets up a status page that you can use to see what the server is doing. You run the **ffserver** as a daemon **ffserver &**

Now, you use **ffmpeg** to provide a feed. We'll use our screencast example to provide a live stream of our desktop (you might instead stream from a webcam).

```
ffmpeg -f x11grab -r 25 -s 1024x768 -i $DISPLAY http://localhost/feed.ffm
```

You can then point a web browser at **http://localhost/stream.swf** to see the live streaming video or at **http://localhost/status.html** to view the status of **ffserver**.

With Mencoder and FFmpeg, you can do just about anything you can imagine with multimedia files. They are complicated tools, but they're both well worth making the effort to learn.

“MPlayer and FFmpeg are complicated, but are well worth making the effort to learn.”

John Lane is a technology consultant with a penchant for Linux. He helps new business start-ups make the most of open source.

/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!

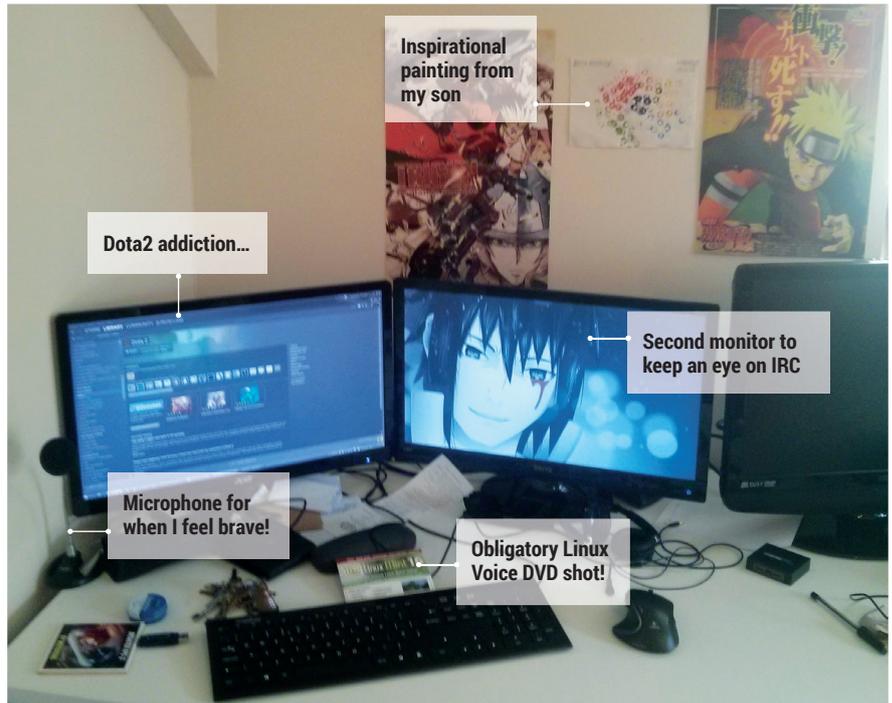
A short time ago I found myself at the head of a table one of the many restaurants of a large Las Vegas hotel. To the left of me, some of the very best Go programmers in the world were conducting a vociferous and sometimes vicious argument with those situated opposite, a collection of some of the world's very finest Python programmers (these are not mutually exclusive sets, but it paints a nicer picture).

As I slurped on my noodles, I listened in wonder and some small degree of awe as the fundamentals of each language were subject to the sort of withering critique only possible if the participants have spent much of their lives worrying about integer sizes and garbage collection. There was much to-ing and fro-ing on DRY principles and boilerplate code.

Eventually, after slipping a small but exquisite rum baba down the hatch, I noticed attention had turned in my direction, as though having exhausted rational argument some sort of consensus could only be agreed by an independent arbitrator.

"You are all wrong" I said, daintily dabbing crumbs from my lower lip, "everything since 6502 machine code was a mistake".

I do have a weakness for the elegance of some well turned assembly, but it is true that, though ruthlessly efficient, it is rather hard to express some of the modern world's problems easily in the language of machines. The particular problem they were grappling with was concurrency, and it is indeed something that Go does more convincingly than Python. (Cue flame war.) But maybe that's just because this is the best book-burning description of concurrency I have ever seen: <http://concur.rspace.googlecode.com/hg/talk/concur.html>.



My Linux setup **Liam Dawe**

Originator of gamingonlinux.com – and our games editor!

Q What version of Linux are you using at the moment?

A Right now I'm using Linux Mint 16 on the desktop, as I find it comes with everything you need. I originally used Ubuntu, but I just don't like their direction anymore. I also use Android on my Nexus 4 as there isn't a Linux distro for phones I am comfortable with putting on it yet.

Q And what desktop? Cinnamon or Mate?

A Cinnamon 2.0. It has the best mix I've found between simplicity and features while not being KDE-flashy and in your face.

Q What was the first Linux setup you ever used?

A My first taste of the Linux brew was actually Mandrake 9.2 before it became Mandriva – the first PC that was

purchased for me actually came with it pre-installed and I could never keep myself away from Linux since then.

Q What Free Software/open source can't you live without?

A It has to be Simple Screen Recorder, as it's hands-down the best screen capture software around. Many people just use some sort of FFMPEG script that just crams everything audio/video together without making it sync correctly, where as SSR does this for you.

Q What do other people love but you can't get on with?

A Gnome Shell & Elementary. Give me back my darned minimise buttons! Elementary at least has a normal dock where applications minimise down to, but Gnome Shell is just trying too hard to be different. ☹

LINUX VOICE ISSUE 2 COMPETITION RESULTS

They are the elect to whom beautiful things mean only beauty. For the rest of us, beauty is Python



In issue two, we challenged you lot to draw pictures using just 100 lines of Python and the turtle module. The results were so fantastic that it was too difficult to pick just one, so we've decided to take the cowards' way and have a joint first place. Both entries used clever maths and clever programming, and they used them to create very different pictures.

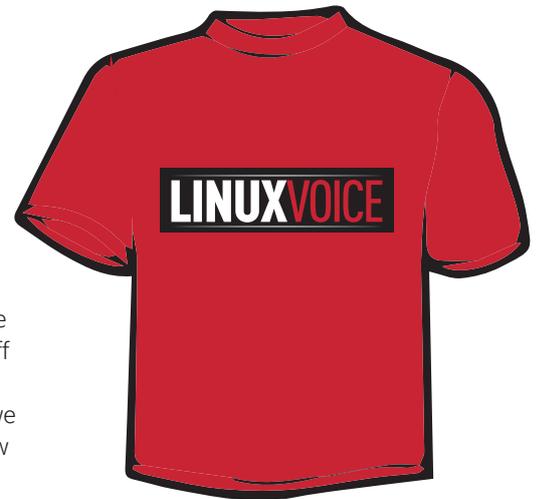
We'll put all the entries (along with the source code) online at www.linuxvoice.com/python-gallery, so head down there to see all the entries and to take a peek behind the scenes. Although the competition is now closed (and there are no more prizes), we'll keep the website updated with any new pictures that are sent in. To get on the site, send your code to ben@linuxvoice.com.

Just remember the simple rules: no more than 100 lines, and only the turtle module can be used.

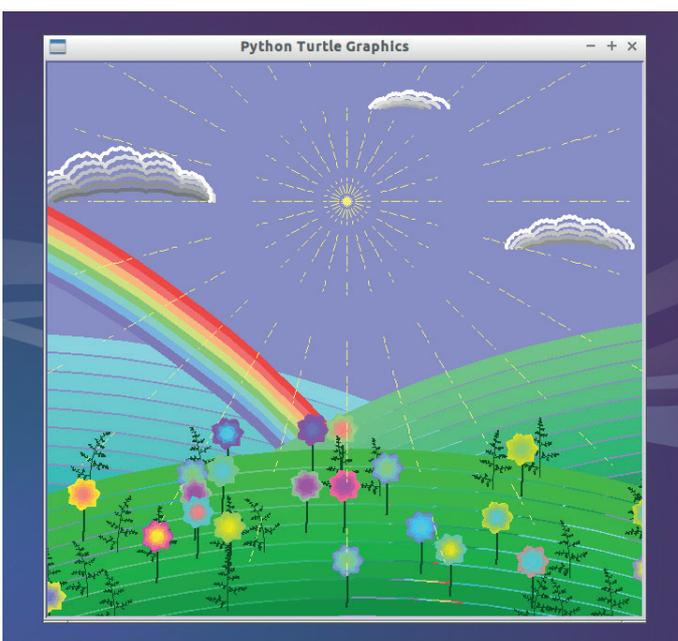
The winners!

Steven Minton's pastoral scene uses the relationship between maths and real life to create a country scene that's far more detailed than we thought would be possible in just 100 lines. The second just shows off the inherent beauty of maths to create a quartet of abstract pieces. Both of them, we feel deserve a place in a gallery, but for now a page in Linux Voice will have to do.

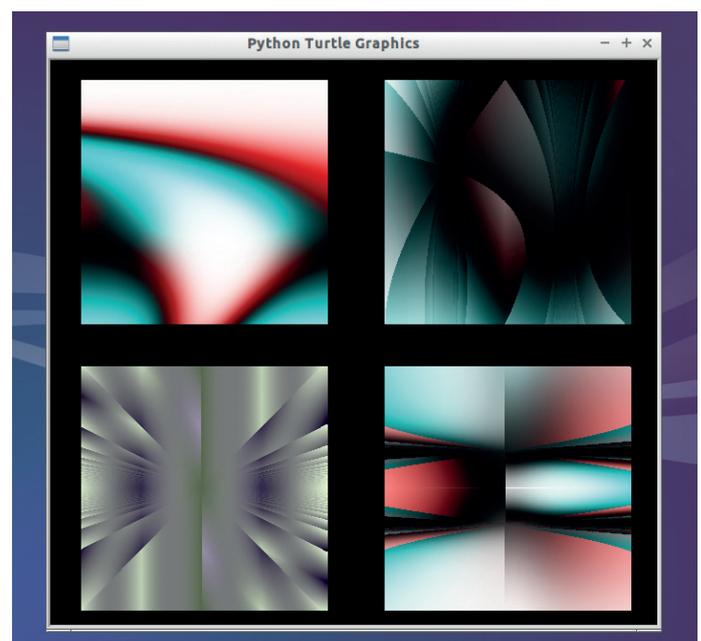
Thanks to everyone who's helped to make the Linux Voice competitions such a great success. Keep an eye out in future issues as we have some big plans!



This could be yours! The exclusive Linux Voice leet T-shirt is only available to the winners of the Linux Voice challenges.



Congratulations to Steven Minton for creating this scene. The clouds and ferns are drawn using fractals, and loops draw the flowers.



There's lots of maths behind this entry. We'd love to tell you how Jonathan Whitaker created this masterpiece, but we haven't a clue.

