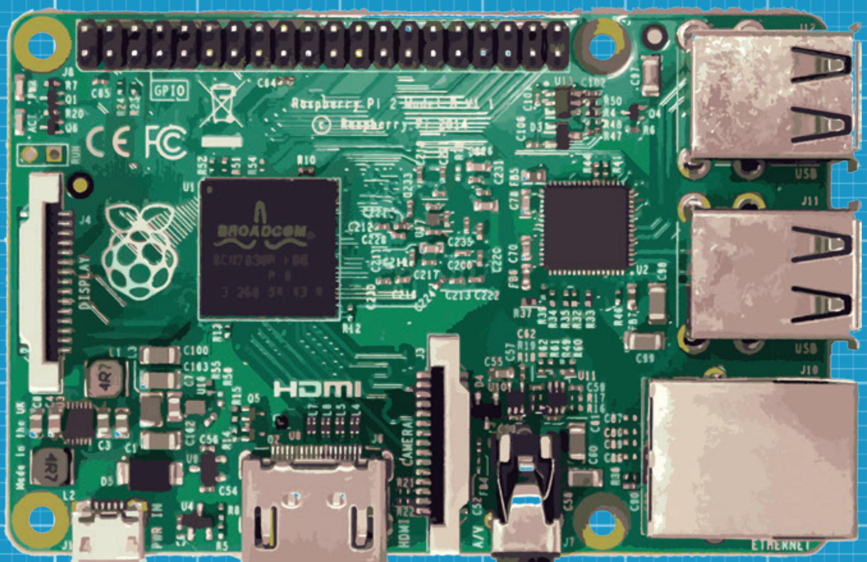
THE BIGGEST AND BEST LINUX MAGAZINE IN THE WORLD

LINUXVOICE

April 2015

FREE SOFTWARE | FREE SPEECH

Raspberry Pi 2



Same GPIO pins – all your robots are still belong to Pi
Double the RAM!
Still only $35
Six times the processing power of your old Raspberry Pi
Still made in Wales/ dal a wnaed yng Nghymru
Run Chrome and LibreOffice at the same time!

Why you need the next-generation machine that's changing everything

BRUCE SCHNEIER Cameron plan "completely idiotic"
DDRESCUE When disaster strikes, your data is not lost
PROFIT SHARING Cast your votes to help the FOSS community

GLOBAL XPRIZE
JONO BACON
How to harness a worldwide community of geeks to save the world

FREEDOM FOR PHONES
CYANOGENMOD
Take back control of your phone with the non-evil Android alternative

ISSN 2054-3778
April 2015 £5.99 Printed in the UK

36+ PAGES OF TUTORIALS SQUEEZELITE • ANDROID • DDRESCUE • IRC • PYTHON • LANGUAGES • ARDUINO • QT • ASSEMBLER

ANDREWS & ARNOLD LTD

will make you the

LINE KING

BE THE MASTER OF YOUR OWN TELEPHONE

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

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

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

HAKUNA MATATA

The future is penguin

The April issue

LINUX VOICE

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

1 At the end of each financial year we'll give 50% of our profits to a selection of organisations that support free software, decided by a vote among our readers (that's you).

2 No later than nine months after first publication, we will relicense all of our content under the Creative Commons CC-BY-SA licence, so that old content can still be useful, and can live on even after the magazine has come off the shelves.

3 We're a small company, so we don't have a board of directors or a bunch of shareholders in the City of London to keep happy. The only people that matter to us are the readers.

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.

As I write this, February's frost is still biting the snowdrops and yet 2015 has already been an exciting year for lovers of Free Software. In the UK, we've had politicians expound their usual (and contrived? and wilful? and negligent?) ignorance of technology, with their comments on end-to-end encryption. We've witnessed a new and particularly awesome Raspberry Pi launch (see page 20) and I'm heading off to London tomorrow for the insiders' launch of Canonical's long-awaited and awesome-looking Ubuntu Phone. In the US, the Federal Communications Commission has just backed Net Neutrality after a long fight to ensure the internet remains a level playing field for our data.

Computer technology has obviously been vital for half a century or more. But it finally feels like it's becoming part of our social fabric, and Linux and open source is the facilitator for helping many people achieve what they want to achieve – it's the keystone in the technological arch. It's corny, but it's true.

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 64

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, Andrew Conway, Liam Dawe, Richard Delaney, Marco Fioretti, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Les Pounder, Mayank Sharma.

What's hot in LV#013

MAYANK SHARMA

"I loved Andrew Conway's tutorial on using the command line to grab share prices in an attempt to shed light on the markets." **p90**

BEN EVERARD

"Mike was able to meet a longtime hero of mine – Bruce Schneier. It was only brief chat, but truth is indivisible." **p28**

MIKE SAUNDERS

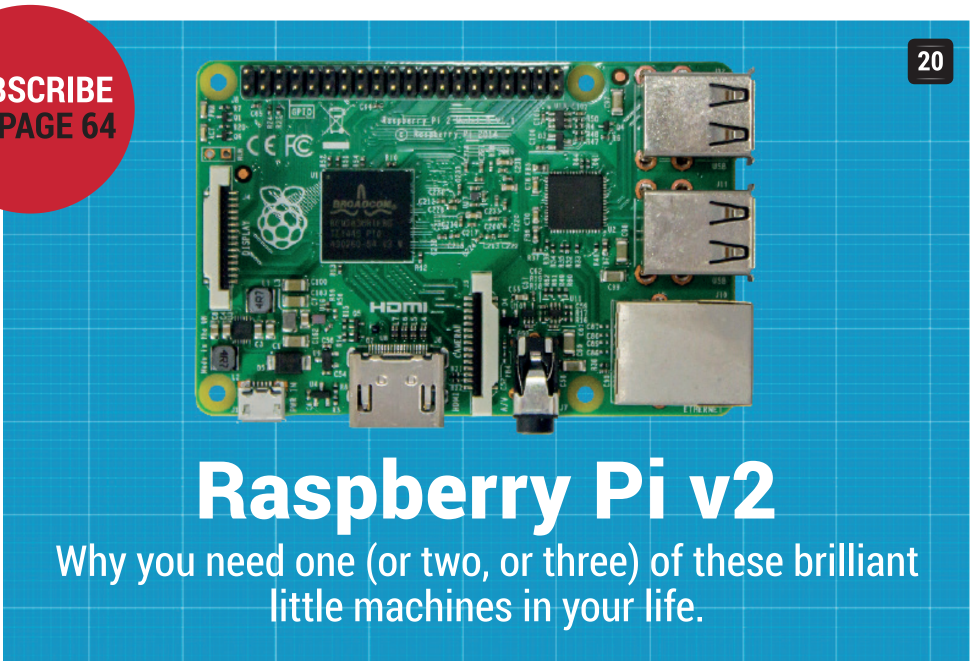
"Ben does his usual exceptional job at dissecting the new Raspberry Pi hardware – I can't wait to get mine!" **p20**

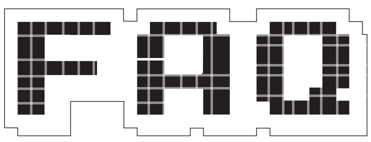
CONTENTS

April LV013

Welcome to 114 pages of Linux and Free Software goodness.

SUBSCRIBE ON PAGE 64





CLIPPI THROU
ONE MAD WEEK IN VID
LEIGH ALEXANDER
FREE CULTU
HOW BIG MEDIA USES TECHNOLOGY AND LOCK DOWN CULTURE AND CONTROL CRE
LAWRENCE LESSIG
Raspberry Pi FOR DUMMIES
Sean McManus
Mike Cook

NEWSANALYSIS

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

Opinion

OSCON – a tradition endures

Welcome back to the European side of the Atlantic.

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

The flagship conference of America's open source communities is holding its 16th event this year. Back in 1997, Tim O'Reilly was making an impact with his technology books. His core business idea, built over a decade of publishing experience preceded by a decade of consulting, was to find the innovators who were writing the emerging software tools – usually for UNIX and its compadres – and getting them to write the sort of manuals that proprietary vendors would issue with their software. Having been one of the first businesses to benefit from what would become the web bubble, after selling the Global Network Navigator search portal to AOL, O'Reilly Media was ready for growth.

As anyone building a software-related business soon realises, you have to run conferences for your customers to gather. That attracts more people to become customers, as well as gathering people from the wider ecosystem. O'Reilly's growth strategy was thus to start a conference to serve his community. Given the power of programmers to set the agenda in the emerging web industry, it was natural to focus the event on the powerhouse of the LAMP stack –including the Perl programming language. Perl's creator Larry Wall became an institution with his State Of The Onion speeches.

From Perl to open source

In 1998 a gathering of people convened by O'Reilly and concerned about helping businesses benefit from free software coined the term "open source" to avoid the ambiguous word "free" and describe the pragmatic embrace of software freedom. The term caught hold and in 1999, O'Reilly renamed his conference "The Open Source Convention" – OSCON. Starting out as a California event and then moving to Portland, Oregon, the conference became a fixture in the American open source scene.

I've been attending it since the 2000 edition in Monterey, CA, when Sun released *OpenOffice.org* as open source for the first time. While I've been a keynote speaker more than once as well as a regular session speaker, I attend because of who else is there. It attracts a Who's Who of open source, and the "hallway track" – meeting people casually outside sessions – is the best value destination of open source relationship building outside Europe's enormous FOSDEM event.

OSCON is showing renewed attendance as a new generation of developers shows up to hear about a new generation of technologies. These days, open source is the default rather than an exception, with open source components dominating every mainstream technology stack from bottom to top.

"OSCON is showing renewed attendance as a new generation of developers show up."

Given the O'Reilly focus on programming languages and tools, the conference has traditionally been arranged around them. But the 2015 edition is going to be different. Rather than arranging the event around technology families, the organisers have decided to pivot and organise around aspects of technology life. Rachel Roumeliotis, O'Reilly's new content director and OSCON co-chair, says "our new track system is set to focus on issues that engineers face during a project and in their daily job. This reflects the open source world that we see around us. Rarely is an engineer solely using one language or framework– these days, there are often several great tools for the job, including SaaS, PaaS, proprietary and open source offerings."

Fresh start

The content at OSCON comes from community experts and has historically been much more diverse than would be expected at conferences run by technology vendors. Traditionally the paper selection process has involved a large volunteer committee vetting proposals and then the conference co-chairs shaping the final agenda based on their input. That has often left a feeling that the content is somewhat random. But this year, O'Reilly is introducing the concept of track chairs. Roumeliotis reckons "Track chairing will involve managing the reviewing for that track, setting the tone and schedule for the day or half day, and working to ensure that an attendee would be able to stay with one track from beginning to end."

OSCON in 2015 is beginning to sound like a conference reborn. More than that, OSCON Europe is back too, in Amsterdam at the end of October. It's good to see that some traditions are durable!

Tizen • Laptops • Systemd • Compute Stick • LibreOffice for Android • Gnome

CATCHUP

Summarised: the biggest news stories from the last month

1 Samsung launches its first Tizen smartphone

Samsung's Android-based devices have largely been market successes, but the company isn't hedging all its bets on a single OS. It has also been working on Tizen, another Linux-based platform, and has now announced the first mobile phone to ship it. The Z1 is only available in India right now, has a 1.2GHz dual-core CPU, 768MB RAM and a 4-inch screen, and costs a smidgen over £60. So it's not a powerhouse, but an entry-level model. **http://tinyurl.com/samsungz1**

2 Big vulnerability in Glibc gethostbyname routine

After Heartbleed and Shellshock, it seems that all high-impact security vulnerabilities now need catchy nicknames. "Ghost" is a buffer overflow in the GNU C library's gethostbyname routine, which is used by a huge number of programs and could potentially lead to arbitrary code execution. Strangely enough, this was actually fixed in the Glibc source code in 2013, but it wasn't marked as a security fix and therefore not backported to older long-term-support distros.

3 Dell announces M3800 laptop running Ubuntu

If the XPS 13 isn't beefy enough for you, Dell has announced a more powerful "mobile workstation" with a 15-inch screen, Core i7 chip and up to 16GB RAM. We're being sent one – stay tuned. **http://tinyurl.com/kqmcvy6**

4 Intel Compute Stick turns TVs into Linux boxes

It's four inches long, plugs into a TV's HDMI port, and contains pretty much everything you need in a basic computer: a quad-core Atom CPU, 1GB of RAM, 8GB of on-board storage, a micro-SD slot, Wi-Fi and a USB port. Not bad for $89. Intel is marketing these gizmos at consumers (eg for browsing the web from your couch or watching streaming media services), embedded devices (such as web kiosks) and businesses (thin-client solutions). **http://tinyurl.com/nrmx5n2**

5 Systemd keeps growing; gets a boot loader

If you're not a fan of Systemd, you might find this overkill, but there are technical arguments about linking the bootloader, PID 1 (init system) and basic low-level utilities together for a more cohesive system. Lennart Poettering and Kay Sievers are considering adding the *Gummiboot* loader into Systemd, to "implement the full trust chain from the firmware to the host OS, if SecureBoot is available". Here's more on what's to come in Systemd this year: **http://tinyurl.com/systemd2015**

6 Party like it's 1995: EISA support stays in Linux

A useful discussion came up on the Linux kernel mailing list in January: one developer posted a patch to remove support for EISA, a bus standard for connecting add-on cards that was popular in the late 80s and early 90s. Linus Torvalds rejected the patch, however, stating: "If we actually have a user, and it works, then no, we're not removing EISA support. It's not like it hurts us or is in some way fundamentally broken". So there's the proof: Linus can indeed be nice.

7 LibreOffice Viewer available for Android

It's currently in beta, and it can't do much aside from looking at basic documents right now, but *LibreOffice* is gradually heading to our mobile devices. Support for more complex documents – along with editing facilities – will follow. It's available as a free download in the Google Play store, and is a joint effort between Collabera and Smoose BV. Early user reviews have been positive, despite the inevitable bugs. **http://tinyurl.com/lo4android**

What is LibreOffice?

Do more – easily, quickly

LibreOffice is a powerful office suite; its clean interface and powerful tools let you unleash your creativity and grow your productivity. LibreOffice embeds several applications that make it the most powerful Free & Open Source Office suite on the market: Writer, the word processor, Calc, the spreadsheet application, Impress, the presentation engine, Draw, our drawing and flowcharting application, Base, our database and database frontend, and Math for editing mathematics.

Finally, documents that look good

Your documents will look professional and clean, regardless of their purpose: a letter, a master thesis, a brochure, financial reports, marketing presentations, technical drawings and diagrams.

8 Gnome (briefly) takes first place in Arch statistics

Which is the most popular desktop among Arch Linux users? Something packed with features like KDE, or minimalist and keyboard-driven like i3, right? Well, Gnome 3 managed to take the top spot for a while, which surprised many due to that desktop's focus on new users and limited customisation options. It certainly got plenty of debates raging on the web. KDE has since grabbed the top slot, but the other stats are worth a look: **www.archlinux.de/?page=FunStatistics**

DISTROHOPPER

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

Evolve OS

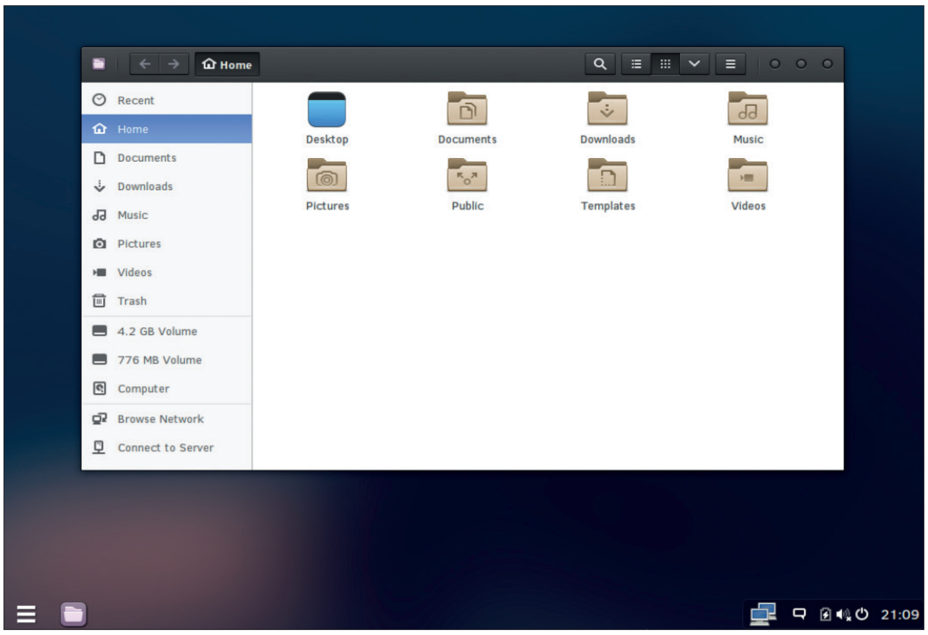
A new distro with a new look.

There are two things that make EvolveOS unique: The desktop and the package manager. The Budgie desktop environment takes the best of *GTK 3* and brings it into one really good looking desktop. If feels like the Budgie developers have taken the good bits of Gnome 3 and Unity and combined them with the simple usability of Gnome 2 to create something wonderful. They haven't recreated all the basic software; instead they rely on the Gnome 3 software stack even for things as integral as the file manager and the terminal.

Our one reservation is about the menu. At the moment, it consists of all your software in a big list with no hierarchy (although it is grouped by type). We tend to have a lot of software installed, and this approach feels like it could get cumbersome quite quickly. Hopefully this is something that will be changeable in the final release.

The package manager is called *eopkg*, and is a fork of *Pisi*, which was developed for Pardus (which has now dropped it in favour of *apt-get*). The idea behind *eopkg* (and *Pisi*) is to build a new package manager that cuts out the cruft that older package managers like *Yum* and *apt-get* have acquired from themselves being wrappers over other systems, leaving a simpler command syntax.

We're rarely excited about a new distro, but we are about Evolve OS. It's still in Beta at the moment, but we're expecting great things from it.



Evolve OS comes from Ikey Doherty, the man behind the excellent SolusOS.

Netrunner

Regular or rolling releases: take your pick.

KDE is a wonderful desktop environment, but it has terrible default settings. That means you either have to find a distro that makes it look nice, or do it yourself. Netrunner comes with one of the best KDE configurations around. The cashew is tucked out of the way and not left to confuse users, the Homerun kicker provides an easy way of launching applications, and the blue and grey colour scheme is inoffensive without being boring. The one thing we're not sure about is the transparent blue window decoration, but at least it's better than the usual blue glow.

Netrunner comes in two versions: the standard version is based on Ubuntu, and Netrunner is sponsored by the same company as Kubuntu (Blue Systems), while Netrunner Rolling is based on Arch (via Manjaro) and is compatible with the Arch User Repository. Which one is best depends on how you feel about rolling releases. Both have the same great KDE interface, but they offer quite different experiences because the standard version is built on Ubuntu LTS, which should remain fairly unchanged for the next four years (unless you want to update it sooner).

Want a good-looking KDE version without spending hours configuring it yourself? Netrunner could be for you.

The rolling release, on the other hand, will constantly chase the latest software giving you a more up-to-date system, but with a higher potential for breaking things. The choice, as they say, is yours.

Bodhi 3.0.0

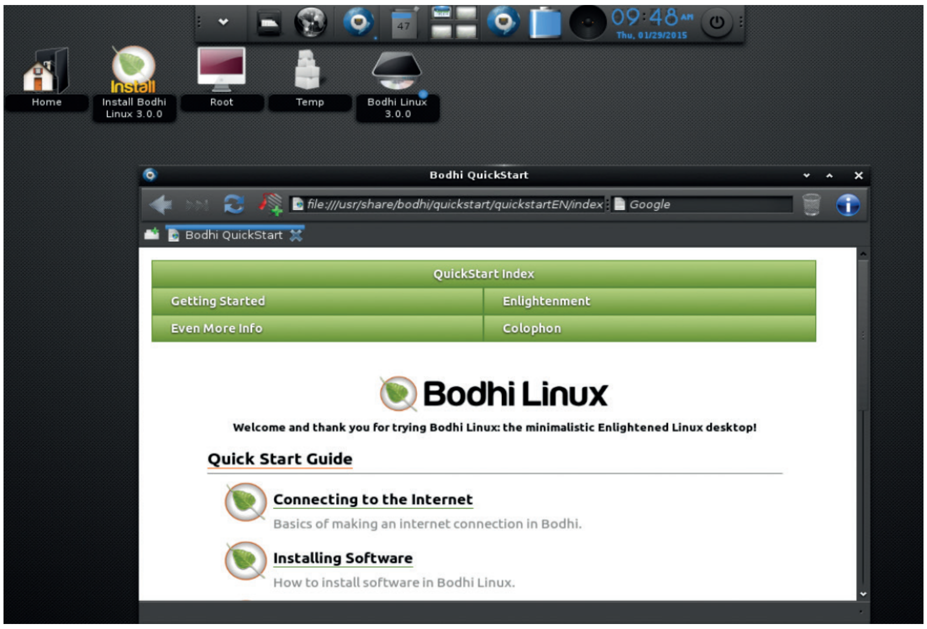
In search of Enlightenment.

In September 2014, Jeff Hoogland stepped down from his role leading Bodhi Linux. Others offered to take on the job, but it seems that he just couldn't stay away. Jeff's back at the helm of the project, and version three is on the way. The second release candidate of Bodhi 3 is out, so we have a chance to see what's in store.

Bodhi is based on the Enlightenment desktop, so looks unlike almost any other distro out there. It's particularly well suited to lower-specced computers, but still comes with graphical flourishes. There's a legacy version designed to support older computers, which is designed to work with machines too old to support the PAE processor feature. The legacy version also has a slightly older version of Enlightenment.

Perhaps the most unusual thing about Bodhi is the web-based App Centre, which enables you to install software straight from your web browser. Actually, it's not that unusual. It comes to Bodhi from Ubuntu and most Ubuntu derivatives have it. There's also a similar feature in OpenSuse. The problem is that users of these distros have never really taken to the idea of installing software through a web browser, so we're looking forward to seeing whether Bodhi can make this work where others have failed.

We're glad to see Bodhi back, and Jeff Hoogland at the helm. It's a great distro, and the Linux world would be a poorer place without it.


Home
Install Bodhi Linux 3.0.0
Root
Temp
Bodhi Linux 3.0.0
09:48 AM
Thu, 01/29/2015
Bodhi QuickStart
file:///usr/share/bodhi/quickstart/quickstartEN/index
Google
QuickStart Index
Getting Started
Enlightenment
Even More Info
Colophon
Bodhi Linux
Welcome and thank you for trying Bodhi Linux: the minimalistic Enlightened Linux desktop!
Quick Start Guide
Connecting to the Internet
Basics of making an internet connection in Bodhi.
Installing Software
How to install software in Bodhi Linux.

Don't let the eye candy fool you. Bodhi really does run well on older hardware.

Ubuntu 4.10 The birth of a new age

Back in the early years of the new millennium, Linux was a complex beast to use. Things broke – a lot – it fixing them wasn't easy. Even installing most distros required more knowledge about your computer than the average user had. One man set out to change this, and create "Linux for human beings". That man was Mark Shuttleworth, and the distro was Ubuntu.

Warty Warthog, the first version, released in October 2004, came as both a live CD and an installable distro. At the time, that was quite a rare thing. There were live distros (such as Knoppix and Demo Linux), and there were normal distros, but bringing the two together meant you could try out Linux, then install exactly the same thing.

The earthy tones of the colour scheme may not have been to everyone's taste, but the Gnome 2 desktop worked well, and the publicity the distro received attracted many people from outside the Linux world. We meet many people today who trace their Linux using back to Warty Warthog.

These days, Ubuntu is best known for the controversies over Mir and Unity, but these are blips in its 10-year history. Ubuntu was – and still is – Linux for regular people who want a distro to just work. By focussing on this aspect, it changed the trajectory of desktop Linux as a whole and made it more accessible. For this it should be commended. We look forward to another 10 years.

In hindsight, shades of brown may not have been the most flattering colour scheme, but they did make Linux look less threatening.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

R.I.P. ETHELRED

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

OpenGL has been getting a lot of flak from some prominent names, and a lot of people have agreed that something needs to be done, and soon. Luckily, Khronos Group, who oversee OpenGL development, have been listening, and glNext will be unveiled this year at the Game Developer Conference.

The major problems with OpenGL include vastly different driver support across manufacturers like AMD, Nvidia and Intel. AMD graphics card users will have seen this the most, with the official Catalyst driver often performing far worse than Nvidia across many games.

Luckily, glNext already has backing from big companies like Valve, Unity and Electronic Arts, so we could see some bigger games coming to Linux if companies like EA are already interested in the new API.

For gamers, glNext should mean better performance, and that's what we all really care about, right? We have some serious competition from Microsoft with DirectX12 and AMD with their Mantle API, so OpenGL has a lot of catching up to do with glNext, and plenty of developers hearts to win over.

The problem with glNext is that it needs driver support from the big three (AMD, Nvidia and Intel), and it's not likely they will update a lot of their older cards to support it, so it will take a long time before it becomes useful. It probably won't even be ready this year, but we're keeping an eye out.

http://forums.linuxvoice.com

Planetary Annihilation

Strategy on the grandest scale we've ever seen.

Planetary Annihilation is a game we've covered before, albeit quite briefly, and since we last looked at it the real-time-strategy game has come on in leaps and bounds compared to the earlier releases.

This is the biggest strategy game Linux has, and now that we have been able to spend a lot of hours with it we can safely recommend it for strategy fans, as to put it simply it's "smashing". If you don't get that pun here's a tip: you can smash entire moons into planets by building massive engines on them, hence the use of the word "smashing".

One of the most recent updates added a building that fires units from one planet to another, giving the game another possible strategy element.

It's not for gamers looking to hop in and out, as the average game will last at least half an hour for anyone reasonably good at it. That's actually one of the highlights, as you get a lot of time to plan your attack: are you going to conquer by land, sea, air or space?

Another highlight of the game is the minimap, which is essentially the game being played again in a small window that you can move around, and you can switch your main view to where you're looking on the minimap at any time.

There's a lot to like about *Planetary Annihilation*, with multiplayer lobbies enabling you to easily join games with other players, and ranked matchmaking to test your mettle against other players, and gain places in the leaderboards. It turns out we are pretty pants at the game compared to most people. There are a couple of rough edges, but it's being polished up to be a good experience, as with our testing we didn't find too many crashes for the Linux version at all.

You will also need a fairly decent gaming rig to get full enjoyment out of it, as even on one of our best rigs the game became increasingly sluggish as more units and buildings are created on each side, and the problem gets worse with more players.

Store http://store.steampowered.com/app/233250

Is that the death star? Oh god it's firing at me!

"You get a lot of time to plan your attack: are you going to conquer by land, sea, air or space?"

Satellite Reign

The spiritual successor to Syndicate.

We have very fond memories of playing the original *Syndicate* game on the Amiga, so this is a bit like walking down nostalgia lane, but with prettier graphics and updated game-play.

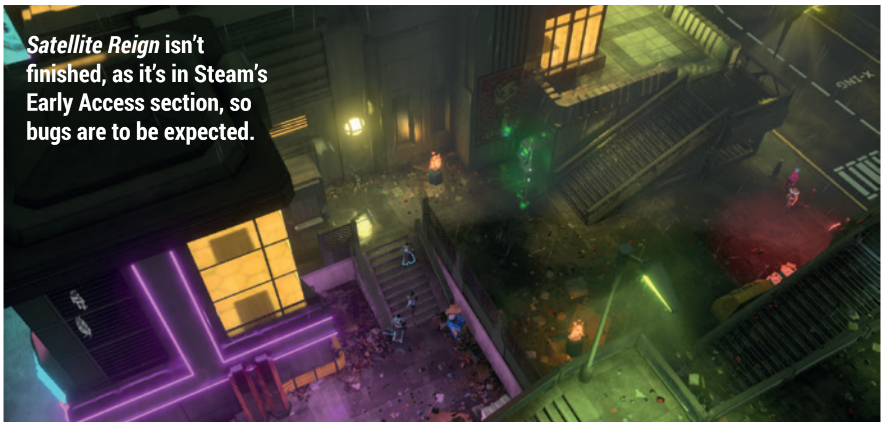
It's a real-time strategy game played with a team of four different agents with unique abilities, and it looks absolutely fantastic already. The graphics are vibrant, and any fans of cyberpunk-styled games and art will probably love it at first sight.

You can customise your team with different abilities, and this is what makes it really fun, as with many different combinations each play-through will be different.

It's played from a top-down perspective, and anyone who has played class-based strategy games before should find it pretty easy to manage. If you've played *Wasteland 2*, *XCOM* or anything like that, it should seem pretty familiar.

The developers have been quick to pump out new versions with new features and bugfixes, so we feel it's pretty safe to pick up and play now.

Store http://store.steampowered.com/app/268870


Satellite Reign isn't finished, as it's in Steam's Early Access section, so bugs are to be expected.

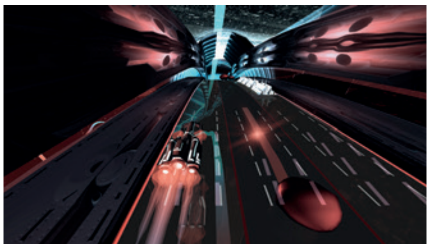
Audiosurf 2

Music and Gaming fused together perfectly!

Sadly, Linux never gained a port of the original *Audiosurf*, which is arguably one of the best music-orientated games around. Luckily, the second iteration has been ported over recently, so you can get your groove on to your favourite tunes in this arcade game.

We can't really compare this to anything else, as there isn't much out there like it – which is one of the things we love about it. The original was very unique and this sequel carries that spirit on perfectly.

It may look a lot like a racing game, but it's more like a single-player time trial that goes with the flow of the current music track you're listening to in the game. It's not exactly easier either, as we found out when playing some rather fast music and it gets a little intense when you're trying to dodge spikes and the game speeds up with the beats of the music.

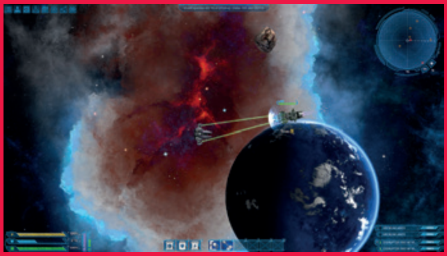


One of the truly great things about *Audiosurf 2* is that you can discover new music with it, as there is a feature where they highlight a specific song each day, and doing this ourselves we have found a few new loves and frustrations.

It's a little on the rough side at times, and this goes with it being yet another game in Steam's Early Access section.

Website http://store.steampowered.com/app/235800

ALSO RELEASED...

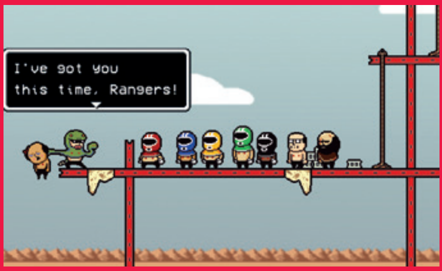


VoidExpanse

Do you miss playing *Eve Online* from your days as a Windows drone? Well miss it no longer! *VoidExpanse* is a great-looking 2D single-player and multi-player space sandbox game, that feels a little bit like *Eve Online*. It's a bit more fun too!

You can fight, trade and mine your way through the game, everything you would expect in a sandbox space game.

http://voidexpanse.com



LISA

We know you love your quirky platformer games, and *LISA* is one of the quirkiest we've come across. It's a side-scrolling RPG set in a bleak world, with a bleak character.

The opening scene sets the tone of the game perfectly, and the small nod to the power rangers in one scene was highly amusing.

http://store.steampowered.com/app/335670

Desura

Desura is the online game store that has been through a few different owners, and the new proprietors seem like a really good bunch.

Desura's desktop client is once again open source under the GPL licence, and they have released a Linux beta. It's a little on the buggy side right now, but it's good to see the newer owners of Desura try to please us Linux gamers.

http://www.desura.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

MANY GOOD IDEAS

I just want to write that I really appreciate the magazine, I enjoy reading the Core Technology section as well as the community-related topics, interviews and tutorials. I like the informal style of your writing and the broad range of topics – good job guys.

For quite some time I wanted to write on a few of my ideas for the magazine – mostly some topics you could cover in the future issues.

1 It would be nice to have a newbie-box in the tutorials to get a clear distinction where is the introductory part and where is the main topic and to give additional information to newbies – eg on the one hand I know a thing or two about Python programming so I could happily skip the first part of the article on Mandelbrot sets. On the other hand I am not that good at electronics so I sometimes miss a thorough explanation why do you use specific resistors or wiring.

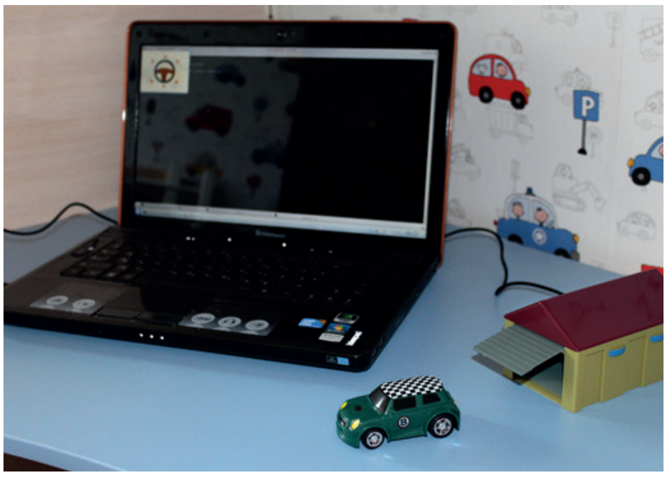
2 Give us a fishing rod, not only a fish! I liked the article about the USB-car controller because, apart from showing an example of reverse engineering, it also explained a little bit of the USB protocol so one can try some other USB devices – way to go! So maybe some further tutorials could cover, for example, I2C and other general-purpose protocols/applications?

3 How to free your hardware? I'm very interested in replacing software in various devices with free alternatives. I know about OpenWrt, but how about putting Free Software on to a TV set, car, fridge, house-entry system etc? A series on setting free different hardware would be nice.

4 How to run a LUG? – I found that the LUG that used to run in my city is effectively dead, so I'm wondering how it could be organised, how to make the first meetings and so on.

5 How to talk about Free Software to people that are not technically-oriented? I've been recently amazed by what Tim Bray said when interviewed by Graham Morrison: "...I don't think we should expect the general population to have educated opinions about internet safety any more than they have educated opinions about emission control systems in automobiles ..." – I realised that this is also true with regard do many other areas of IT, maybe also with regard to the computing as a whole. So how to talk about Free Software to people who don't know what software is?

Dawid Grzegorz Węckowski


Once you've written a driver for one USB device, you pretty much know how to writer drivers for all sorts of other hardware.

PS The word 'free' is ambiguous not only in English. In Polish we've got the word 'wolne' which can mean 'free' (as in freedom) but it can also mean 'slow' :) So there is another level of difficulty when talking about Free Software :)

Andrew says: Wow, thanks Dawid! We can all have a rest from thinking up ideas for a couple of months thanks to you. Sorry for cutting your letter short but there's a limit to how much of your excellence we can fit in!

With regard to flagging up the newbie and advanced sections of tutorials, I'm not sure this is practical, as one man's easy is another man's difficult. The best we can do is make sure we include as diverse a range of topics as possible, which is exactly what we're trying to do.

There's definitely more security stuff on the Linux Voice radar, not least next issue's cover feature, which will be a start-to-finish look at how to and we've always wanted to include more on the Linux kernel. I should get in touch with Dr Sinitsyn (who wrote the USB control tutorial) and find out if he's in the mood to write some more on the kernel. And to everyone else: let us know what you want to see more of, and we'll do our best to provide it.

The community aspect is another great idea – as it happens we do know an expert (who we interview on page 40) – I'm sure we can get something useful together for you in the near future.

SHADOWS IN THE DARK

I just listened to the latest Linux Voice podcast today, and listened with interest to your discussion of David Cameron's call for the elimination of encryption on the internet. While you all made very valid points, I think you missed the mark on what they're really up to.

Obviously the politicos don't want to eliminate encryption on the net. That would stop all net/web-based commerce, and would lead to a loss of tax revenue (VAT, sales, wages, etc) and a potential collapse of the high-tech sector. But the politicos also want to read everything we send over the web to maintain their power. So how are these two things reconciled? Enter the Hegelian Dialectic.

The Hegelian Dialectic is as follows: Problem, Antithesis, Synthesis. As Cameron stated, the government can't see everything on the net but needs to under the guise of the fight against terrorism (the Problem).

Cameron stands up in front of everybody and says he wants to eliminate all encryption, the worst possible solution to solve the problem (Antithesis). The public say they want to maintain their encryption to keep out the evil hackers and keep shopping on Amazon. Parliament/Congress then passes a law requiring everyone to register their private encryption keys with the government, or else their communications will be treated as an act of terrorism and prosecuted (Synthesis).

Lost in all of this is the question of whether the government should have free and clear access to anybody's communications (the public is easily distracted and will lose sight of the actual question – similar to what you were saying about how the politicos have changed the role of the government before our very eyes).

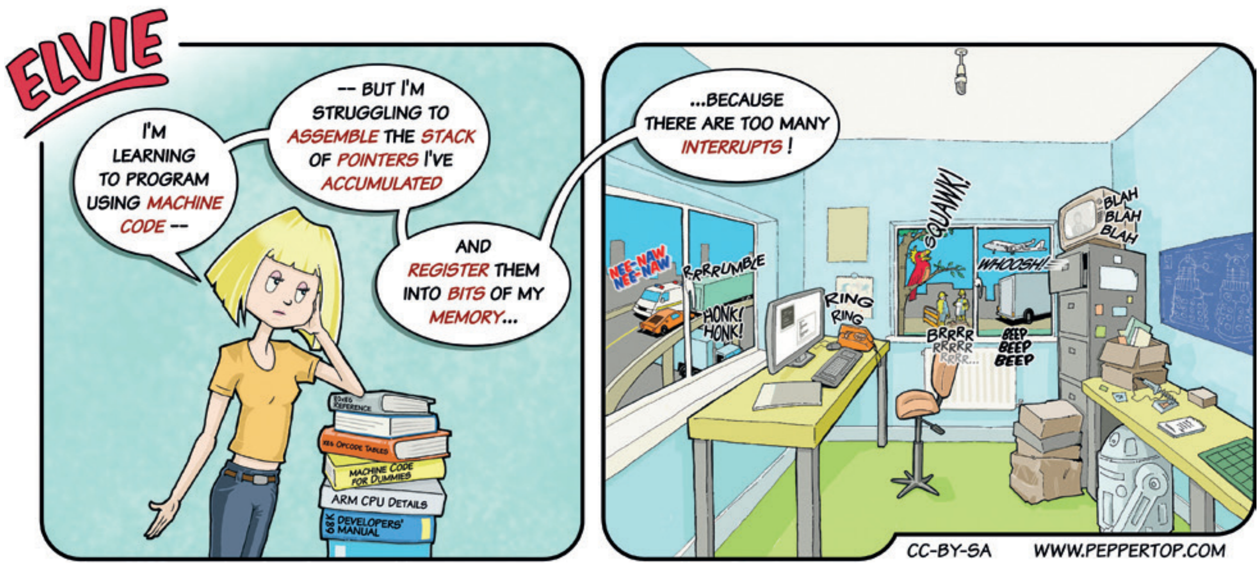
I don't know about the UK government, but the feds here in the US have floated the idea of government-run private key registries for years, which would give them access to all of our communications. This is what they're ultimately after and what must ultimately be resisted.

Keep up the good work,
Paul Olson, Oklahoma USA

Andrew says: Ooh, this is brilliant. I can feel a dystopian novel brewing somewhere. "The public is easily distracted" is brilliantly understated and completely accurate, and your idea of a legally enforced registry of encryption keys is wearyingly plausible. However, I think you give our lords and masters too much credit. During a debate in January in the House of Lords, Lord King of Bridgewater admitted that "I'm not a tweeter. But we've got Facebook, we've got Twitter. The other day, somebody tried to explain to me what WhatsApp is. Somebody tried to explain to me about Snapchat. But, my Lords, I don't know about them. What is absolutely clear is that the terrorists and jihadists do." In other words: "I don't understand what we're talking about, but I want to ban it anyway because terrorism".

It's easy to caricature the Lords as a place where out-of-touch establishment figures go for a nap after a nice taxpayer-subsidised lunch, but Lord King is a former Secretary of State for Defence. Scary!

If in doubt, ban it.



HAPPY BIRTHDAY TO US!

First, congratulations on your first year looking forward to many more Linux goodies from the Voice stable over the coming years. I started using Linux 8 years ago to enable me to give away fully working older PC's that I had refurbished from donated parts on Freecycle (now freegle).

The reason for using Linux was that all donated items have to be legal and many of the PC's did not have a valid windows licence or had originally run windows 98/2000 (or horror of horrors, ME) So if I wanted to give away a working PC ready to use it had to have a Free/Libre OS on board.

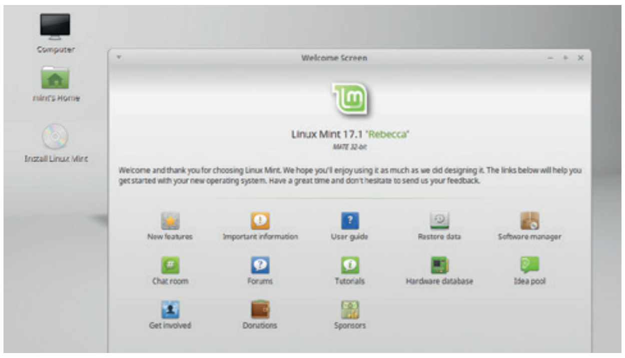
I've recently given away five PCs with Mint Mate on them, and apart from forgetting to inform them of the password on a couple of occasions it seems that they are getting on fine. I also put the Mint user guide on the desktop when installing the PC and tell everyone they can email me if they have any questions. Apart from the password questions, I've had zero calls for support. I have had emails thanking me for the PC my son/daughter loves it kind of thing.

The best was yesterday when a woman I gave a PC tower to a couple of days previously, emailed me to ask if Linux could be installed on her daughter's laptop. Quote: "As she is getting fed up of Windows updates that don't work."

Most people don't care what OS they're using as long as it works for them and does what they want it to, so as long as we give them a system they can use; for some this will be Windows-like, as that is what they are used to and fits the hardware they are using. For others with touchscreens and portable devises this will be Android or Unity as this works for those devices.

Tony Hughes, Blackpool LUG/ Makerspace

Mike says: You're a hero for spreading Linux to the good people of Blackpool, but I wonder if anyone else has had a similar experience? Is Mate really the best desktop for new Linux users, or has anyone else had a good experience with Unity – or even a properly configured KDE?



Welcome and thank you for choosing Linux Mint. We hope you'll enjoy using it as much as we did designing it. The links below will help you get started with your new operating system. Have a great time and don't hesitate to send us your feedback.

Sometimes the old-fashioned way is the best. The Mate desktop is the perfect example of this.

~~DISCOVERY~~ FIND OF THE ISSUE

I found out yesterday (January 25, 2015, Pacific Standard Time) that typing

```
$ cat <<'EOF' | tee input.sh | bash
> lines
> of
> script
> EOF
```

will cause the script reading "lines\nof\nscript" to be executed.

This way, it can be possible to execute the same kind of program you'd read from a book or magazine in the '80s! (Only the program is in Bash's language, not BASIC. input.sh is the location where the script typed would be saved for later execution and editing.)

I have had discussions with Richard M. Stallman, founder of the Free Software Foundation; and with fellow subscribers to the bug-bash mailing list, about this interesting discovery.

Therefore, I suggest that you begin publishing type-in programs, at a rate of no more than twenty-four pages thereof per issue.

I got my inspiration from a July 1984 issue of COMPUTE! Magazine (obtained via the Internet Archive's OpenLibrary Project), which likewise featured type-in programs.

If you (this means readers, too) want to see my discussion with bug-bash, open your Internet browser and enter the URL http://lists.gnu.org/archive/html/bug-bash/2015-01/threads.html. It should be near the top of the list.

Ryan Cunningham

Graham says: We've often discussed among ourselves the anachronism and usability of printing code in 2015, but we all think it's still a good way of digesting ideas and techniques, even if it's not the best way of getting things to run on your computer. That's also why we try and include the code online whenever possible. And I still fondly remember a summer spent typing in the code for the Mystery of Silver Mountain adventure game, only to lose it all when the cassette stopped saving the last save block.

LIBERTY

I totally agree with Simon Phipps in his News analysis piece about the attack on Charlie Hebdo and the rush to make 'security' better by restricting the liberties of everyone.

I am reminded of the following quote which I first saw in a newspaper article shortly after 9/11, but it is a quote from much earlier in the life of what became the USA: "They who would give up an essential liberty for temporary security deserve neither liberty nor security". Benjamin Franklin (1706 – 1790)

John Paton

Simon's right – terrorism must not become an excuse to make us give up essential liberties.

VIRTUAL DESKTOPS

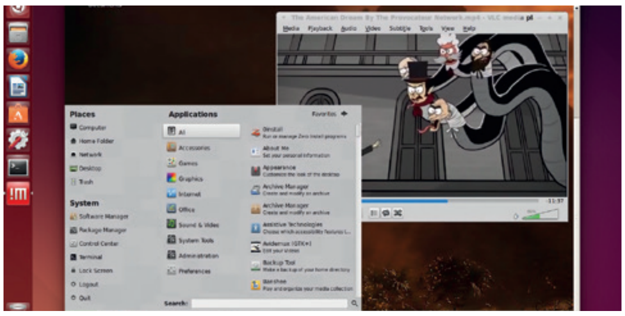
I'm sure you'll get quite a few emails about your Group Test of remote desktop client but somehow you overlooked *X2Go* in your list. A quick search will uncover that *X2Go* is basically a fork of the GPL'ed *NoMachine NX 3.x* libraries with a custom client/server. It is generally used to connect to a virtual desktop but they do offer a "desktop sharing" package to share an existing, physical X11 session.

NoMachine's NX 4.x product is based on a completely new protocol which is, to the best of my knowledge, not "open" (as mentioned under the screenshot on page 63). Version 3.x of the NX libraries is open. Maybe there is a difference between library and protocol? Anyway, perhaps someone would be interested in doing a follow-up for *X2Go*? *X2Go* is included in a few distro stock package repos (Fedora for example) . One problem it does have is that it does not support anything that requires 3D acceleration, so for example, Gnome 3 can't be used over *X2Go*. Other than that, *X2Go* is full of features.

BTW according to the Wikipedia page, *NX 4* can use SSH but, "SSH authentication is available only on enterprise-version servers"... which to me means the free download version of NX 4.x doesn't work over SSH.

Scott Dowdle, Belgrade, MT

Graham says: Thanks for bringing that to our attention Scott, we'll give it a look.



It's proprietary, but *NoMachine NX* was the clear winner of our remote desktops Group Test.

2015: THE YEAR OF LINUX

If only 2015 really were the year of Linux. I worked most of my working life in engineering materials supply and purchasing, and every so often some kind soul in the drawing office would design into our product a component only available from one company, thereby bringing the buyer face to face with his worst nightmare – a monopoly supplier. These companies are always expensive and nearly always have a one-sided approach to dealing with the customer.

For the vast majority of computer users Microsoft has managed to maintain a monopoly. This is partly done by ensuring that we cannot buy a computer on the high street without a Microsoft operating system. I tried to raise a people's petition on the government's website to the effect that shops must offer a choice of operating systems including at least one Linux system; it was refused.

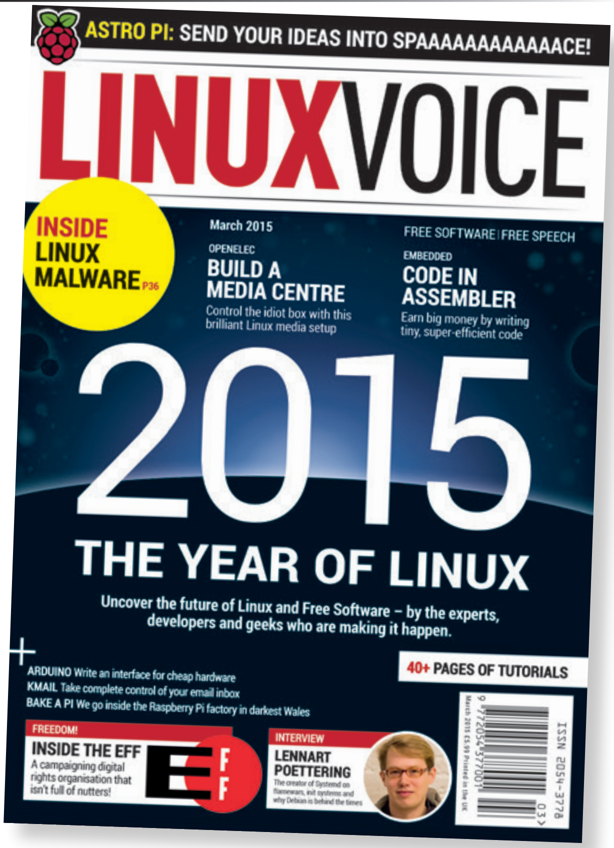
The magazine that I first read about Linux in back in 2010, *Computeractive*, has not mentioned Linux since being bought by Dennis Publishing. A letter to the editor has not been answered. I have written to *Which* magazine asking them to run an article on alternative operating systems, again with no answer. I am not one for conspiracy theories, but I am starting to wonder about the size of Microsoft's slush fund and who is bankrolling it.

The other problem Linux faces is simply a lack of knowledge of the existence of the systems. If you buy a computer new and eventually replace it with a new one you never need to look elsewhere. It's only people who buy second-hand machines and need something better than an obsolete Windows XP system who search for and find Linux. The only bright point is that there are an unknown number of schoolchildren out there using a Linux DVD to get around parental controls. I even came across two young ladies who had managed to hide a Linux OS within the Windows 7 partition and made it invisible until required by reducing the *Grub* menu's countdown to 0.

John Bourne

Graham says: Sometimes we worry about the youth of the nation, but if those girls are anything to go by, the kids are all right. It is, as you say, a constant gripe that there's so litle choice on the high street, but the much-heralded death of the high street is making this less of a worry. The advantages of having a bricks-and-mortar shop are reducing, and as more consumers buy their hardware online, the barriers for entry will fall for the likes of PC Specialist and Zareason (which both offer excellent hardware with Linux pre-installed).

Have faith.



There's a load of great stuff coming for Free Software this year; just don't think that this is "the year of Linux on the desktop".

LUGS ON TOUR

linux.conf.au 2015

Daniel Rossbach reports from this year's event in Auckland.

photos: Simon Lyall CC BY-SA

Each year during winter in the places in which most open source conferences are held - which happen to be in the northern hemisphere - some far warmer and more sunny place in Australia or New Zealand hosts one of the most significant conferences in the Linux world: linux.conf.au (LCA).

This year, from 12–16 January, this place was Auckland in the north of New Zealand, were the conference was staged at the quite flashy facilities of the University of Auckland's Business School. Over those five days, roughly 170 presentations were given and attended by almost 700 Linux professionals and enthusiasts.

As in previous years, LCA managed to attract many prominent personalities as speakers, with Eben Moglen (Executive director of the Software Freedom Law Center), Bob Young (co-founder of Red Hat and chairman of Lulu) and a certain Linus Torvalds delivering keynote addresses.

At the same time, LCA is a gathering of the regional Linux community and has a far less corporate atmosphere and structure than most of the other equally well-known Linux conferences around the world. Organised each year by a team of volunteers, which changes along with the host city and venue, the conference has a familial feel as regular attendees (who make up a large proportion of the audience) meet up and some speaker's talks and surrounding events are traditional fixtures.

Fittingly for a community-centric conference, a defining marker of linux.conf.au 2015 was the range of opinions, interests and styles represented, all of which made for a multi-faceted conference programme, and some controversy sparking discussions.

Linux.conf.au is a refreshing antidote to some of the corporate-sponsored events that take place in the US.

Linux flying

While there were a large number of highly technical presentations on the inner workings of Linux and the (database and virtualisation) server-side software commonly run on it, LCA also offers a forum for projects in the wider world of 'open culture'. Talks and interactive sessions belonging to the later group seemed to be the most engaging events within the conference.

Examples for this at this year's LCA were numerous: The use of free and open software in humanitarian aide was the topic of one of the 10 'miniconfs' that make up the first two days of the schedule, which were given even more prominence.

Political aspects of using FLOSS were discussed in multiple talks concerned with the potential and difficulties associated with its use inside and to interact with governments and bureaucracies. And, unsurprisingly, there were reports on yet more embedded Linux deployments.

One highlight of the week in this vein was an impressive demo in

Andrew Tridgell's presentation entitled "Flying with Linux". In it, he made a small plane fly on a predetermined course and land in one piece – with the plane being hundreds of miles away (in Canberra), and simultaneously compiling the Linux kernel on the BeagleBone Black unit inside the aircraft.

Further fascinating avionics were on show outside of the conference schedule at a rocket-launch organised as one of the many 'birds of a feather' events traditionally surrounding the conference. The largest rockets, launched at the New Zealand Rocketry Association in Taupiri south of Auckland reached an altitude of up to 10 kilometres!

Making things

Actually making things was the centre of two miniconfs, an open-radio session and a robotics workshop. At the latter, attendees were given the opportunity to assemble a robot-vehicle out of a Raspberry Pi (model B+), an Arduino and two servo engines before finally steering it over wireless with Node.js scripts.

Beyond these specialised sections, the main conference programme was not short on interesting hardware projects either, as evident in presentations on OneRNG, a successfully kick-started random number generator (the campaign is still running), deploying Linux in automotive systems, and current developments and hacks in 3D-printing.

The biggest news story to come out of LCA 2015 turned out to be based on Linus' appearance on stage. Instead of giving a keynote speech, Torvalds invited questions from the audience. Reminiscing over a similar occasion twelve years ago, Torvalds was joined on stage by the kernel developers and LCA regulars Andrew Tridgell, Bdale Garbee and Rusty Russell.

While Linus went on to make some interesting remarks on disclosure of security flaws (siding, without naming names, with Google in their recent argument with Microsoft), and documentation for the kernel (it is inherently difficult but could and should be improved), the most quotable and controversial lines were delivered regarding the social interactions in the kernel community.

Challenged by Matthew Garrett on Torvalds' abrasive comments on the kernel mailing list and their potentially chilling effects on people getting involved with Linux, Linus replied that he feels under no obligation to conduct himself more amicably.

For Torvalds, his less than polite comments are solely directed at the technical merits of the subject matter, and otherwise merely expressions of him being "a really unpleasant person".

Torvalds claimed that the task of making the Linux community inclusive and conducive for diversity can be delegated among the people who are influential in the Linux community.

This reply was criticised by many as failing to acknowledge the way in which prominent figures, and Linus first among them, set the tone for interaction in the community. The disappointment with Linus' stance was shared by some of the prominent advocates of a more concerted effort to make the Linux community open for people with a background and identity that isn't mainstream in the community.

One of these advocates is Karen Sandler, former executive director of the Gnome Foundation and still co-organising its Outreach for Women programme.

As Sandler, now in charge of the Software Freedom Conservancy, told Linux Voice in the aftermath of LCA, experiences of sexist behaviour are still common for women at events in the open source community. According to her, prominent figures are called upon to exert their influence in the community to change the climate in such a way as that bad experiences for women or minorities don't occur.

Where this development will lead we will be able to see at the next few editions of LCA: in Geelong in 2016, and Hobart in 2017.

All the talks (**https://linux.conf.au/programme/schedule**) from LCA 2015 are available online at **http://mirror.linux.org.au/pub/linux.conf.au/2015/** and on YouTube.

Judging by the conference talks, Kiwis seem to be obsessed with flight. Oh, the irony.

20% of speakers were women (which is far from parity but on an upward trajectory).

TELL US ABOUT YOUR LUG!

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

The French Ministry of the Interior was one of the event's biggest sponsors, and had a huge stand to show it.

The show floor was mostly filled with shiny company stands trying to woo visitors.

FIC 2015

Cybersecurity experts met in Lille to prepare for upcoming attacks. **Mike Saunders** was there.

Lille is a medium-sized town in northern France, just a croissant's throw away from the Belgian border. It has cobbled streets, decent beer, and for the last few years it has hosted FIC, the International Forum on Cybersecurity. This conference is organised in part by the French Gendarmerie and Interior Ministry, and is funded by the regional government of Nord-Pas de Calais along with various sponsors such as Symantec, IBM and Sophos. And it's serious business: security at the entrance was tight, and there were plenty of well-armed guards keeping tabs on the show.

Some big names in politics were there, such as the Ministers of the Interior of France (Bernard Cazeneuve) and Germany (Thomas de Maizière). We found the latter's presence somewhat ironic, though, given that he recently expressed support for David Cameron's plans to have a back door in all end-to-end encryption. Many experts in computing security have described these plans as ridiculous and impossible to implement, so hopefully de Maizière actually learned something during his visit.

But anyway. On the show floor, the French Ministry of the Interior stand dominated, telling the world how it's on top of cybercrime and security in general. FIC took place only a few weeks after the Charlie Hebdo killings, so issues around monitoring suspected terrorists, encrypted communications and freedom of speech came up with many people we talked to. Most of the show stands were from small or medium sized companies touting cybersecurity wares – usually server-side programs, but some for end-users as well.

Some companies were offering Linux versions of their solutions, although one we chatted to – Titania, a vendor of security auditing software – spoke of the frustration and complexity in dealing with the niggling differences between distros. This is a recurring theme with third-party vendors who don't have their software in distro repositories, and even Linus Torvalds has ranted about it recently, so hopefully there will be some progress in the future.

Challenge time

But our favourite (and by far the geekiest) section of the show floor was the challenge area. Here, teams worked together to perform forensic analysis of systems that had malware installed or other vulnerabilities. This included Android phones that had unrequested software installed, databases with suspicious entries, and USB keys that were trying to spread viruses. A number of tasks were set up (eg find out who is responsible from the logs, or work out what the malware is doing on this phone from a memory dump), and the first team to complete them won glittering prizes including consoles, quadcopters and Raspberry Pis.

The show floor was just one part of FIC though. We attended a few fascinating presentations and talks, such as "How to shut down a botnet". In case you're not aware, a botnet is a group of thousands of cracked computers (almost always running Windows) that are scattered across the world, in homes and in businesses, that are all controlled via back-doors from a single point. They are used by nefarious types in situations where having a large number of IP addresses is beneficial, such as sending spam or performing denial-of-service attacks.

This talk was given by a investigators working for the FBI and the UK National Crime Agency, and described Operation Tovar, an international operation to shut down the Gameover Zeus botnet. You may wonder if law enforcement agencies should be worrying about botnets when there are bigger crimes to solve, but Gameover Zeus had been used to distribute *CryptoLocker*, a piece of "ransomware" for Windows. *CryptoLocker* encrypts a user's personal files, and then demands money to be sent to an anonymous address within a certain time limit – otherwise it will delete them forever. And it has been successful: tens of thousands of people have been affected, and it's estimated that *CryptoLocker*'s creators have netted over £3m.

So taking down the botnet that distributes it was a worthy goal. But also a complicated one, which involved various tricks such as registering 2,600 domain names to stop computers on the botnet from being able to communicate with one another. (For the full technical lowdown on how Gameover Zeus was stopped, see **http://tinyurl.com/kso7fea** – it's heavy going in places, but a good read.)

Head in the clouds

Security expert Bruce Schneier (**www.schneier.com**) arrived in Lille just in time to give his speech in front of a packed auditorium. He described how companies are rushing to outsource their infrastructure – using "cloud" services, hosting email on Google, and so forth. It may seem attractive to hand over the workload to someone else, but ultimately you lose control over your security.

"It's estimated that the creators of CryptoLocker have netted over £3 million."

Schneier also attacked the security software market: "The stuff that wins isn't the best stuff – it's the easiest to use stuff, the cheapest stuff". He referenced prospect theory, and how people are risk adverse when it comes to gains, but risk supportive when it comes to losses. Imagine someone offers you $1,000, or a coin flip where one side gets you $2,000, and the other means you end up with zero. 75% of people are risk averse in this case, and would take the definite $1,000.

But turn this the other way round: you could either lose $1,000, or take a coin flip where one side means a loss of $2,000, and the other side means you break even. Only 25% of people take the definite $1,000 loss here – so people are more risk supportive in this situation. This impacts how people look at security, and how IT middle management types don't want to spend more money on securing their systems.

Schneier characterises the 1990s as the decade of protection, when anti-virus software boomed. The 2000s were the decade of detection, with intrusion detection systems and forensic analysis becoming more widespread, while the 2010s will be the decade of response. (We managed to get a quick interview with Bruce at FIC – see page 28.)

FIC 2015 was a good opportunity for like-minded people to meet and share ideas about security, but there was something odd about a government-sponsored cybersecurity event, given how much they want to spy on us. We can only hope that our overlords have learnt from the talks and presentations, and realise that we won't defeat terrorism simply by giving them our encryption keys.

We're not sure if we were allowed to photograph this, but the army had lots of fancy kit to look at.

Over at the challenge area, geeks had some respite from the corporate tone of the conference and could get some white-hat cracking done.

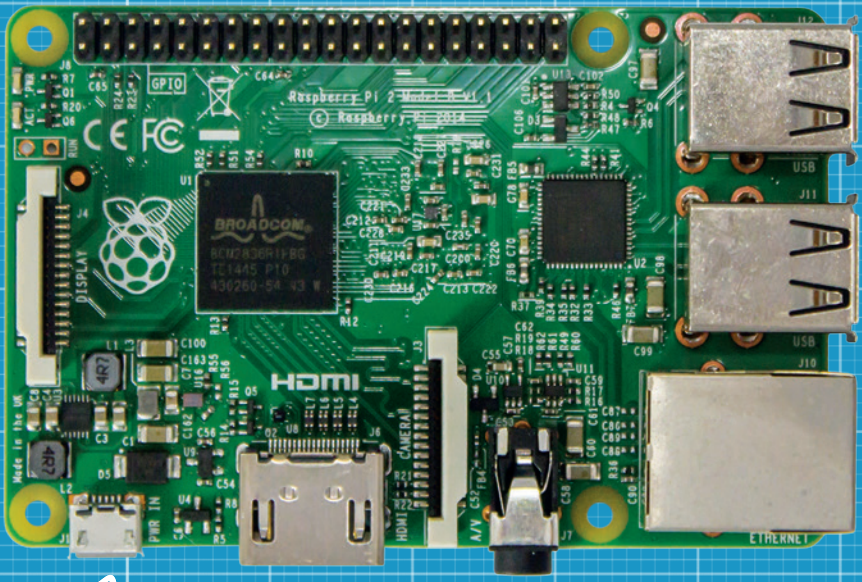
Lille is a pretty place, and Belgian-style beers abound due to that country's proximity.

GPIO pins to control your robot army

Stick it behind your television to make a DIY smart TV

Still proudly made in the land of dragons

Quad-core ARM CPU – lots more processing power



More and faster RAM – replace your old desktop machine no problem

Pocket money price – still only $35!

RASPBERRY PI VERSION 2

The world's most popular computer has a brand-new version. Take a look!

The first Raspberry Pi launched on 29 February 2012. Over the next three years it was used as the brains of a submarine, sent into space, and even used to make delicious beer, but it didn't really change, bar a tweak to the model B+ to improve power handling. This has now changed – Raspberry Pi version 2 is here!

Version 2 is completely compatible with the old board both in terms of hardware and software, and it remains the same price – $35. New computers aren't about what's the same though, they're about what's different. The new board features a new processor and new memory, but it looks almost identical to the older version. What does this mean to the performance, and how will it change the usability? Will it still be used by tinkerers up and down the land, or has the Raspberry Pi Foundation spoilt its creation? Read on for our in-depth exploration...

What has changed?

Don't panic – the Raspberry Pi is the same machine we know and love.

At first glance, version 2 of the Raspberry Pi looks very similar to the Model B+. It has the same inputs and outputs, and an almost identical layout. You'd have to look quite closely to notice the difference, and that's that the RAM is no longer on top of the SoC (System on Chip). It's now on a separate chip mounted on the underside of the PCB.

The reason they're so similar is that the model B+ was designed with this upgrade in mind, and space was left on the board to allow the new features. In fact, if you compare the two devices, you can see that there's a patch of empty space beside the SoC on the B+ that the memory now takes up on version 2.

This similarity goes further than just looks, and the GPIOs are the same. This means that the new board should still work with every circuit and expansion board that's worked with previous versions. Hardware Attached on Top (HATs) should also still all work, and the holes for bolting these onto the Pi are still in the same place.

It's not just the GPIOs that are the same. All the physical connections on the board are in the same place as on the model B+. This means that cases and other enclosures should still work without modification. This particular aspect will be particularly important for developers working on embedded systems.

Eben Upton with a box of the new Raspberry Pis. 200,000 were available on the launch day, and a similar number will be made each month.

So what's new?

The big changes are the SoC and the memory. The first is upgraded to include four ARM Cortex A7 cores, which give quite a bit more power than the previous ARMv6 single-core SoC. The RAM has doubled to 1GB, and is also clocked slightly faster at 450MHz. Combined, these give the board much more processing power.

On the software side of things, the latest version of Raspbian has been updated to support everything out-of-the-box, but other distros may take a little time to catch up. The architecture is now the more common ARMv7 rather than ARMv6-hardfloat. This change means it's much less effort to port other Linux distros to the Pi. This is unlikely to mean we end up with the massive over-abundance of distros like we have on the x86 architecture, but it could mean that a few more of the big names distros produce Raspberry Pi versions.

Video Core IV Understanding the Raspberry Pi's graphics engine

The Raspberry Pi version 2 features exactly the same GPU as the version 1: Broadcom's Video Core IV. Currently, the driver for this is capable of running OpenGL ES programs and decoding video (it can decode H264 and MPEG4 videos at full HD resolutions without problems). You can also purchase additional licences for MPEG2 and VP1 should you need them.

There are also codecs for VP8 and Theora. These run on the GPU, but aren't able to take advantage of the video-playing hardware in the GPU. This means that they are effectively software codecs that happen to run on the GPU rather than the CPU. You should find that playing these files doesn't significantly increase CPU usage, but you may get problems decoding high-definition files. All this is dependent on the compression format.

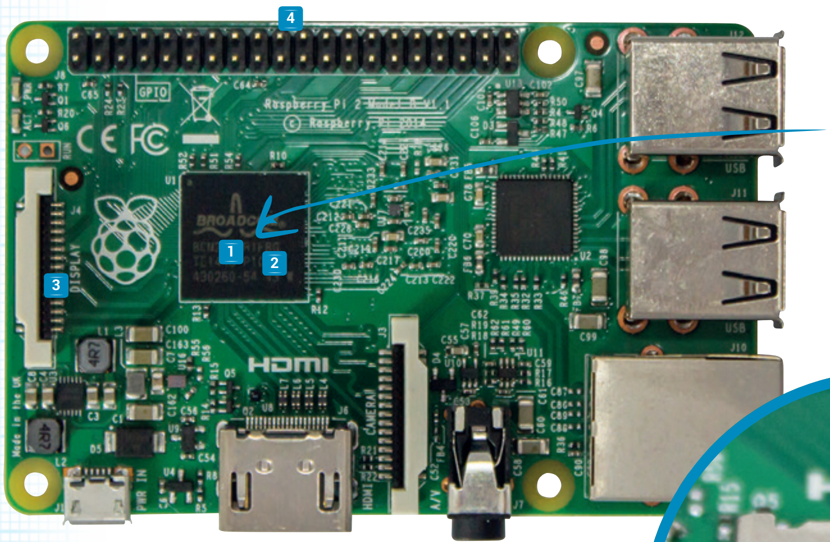
There are also different container formats. The container is the bit that brings the video and audio together into a file. Therefore, the filename corresponds to the container, not the video compression method. If you want to know whether a file will play well on your Pi, you need to know what the underlying compression method is, not what file format it's in.

The Pi can run OpenGL ES 2.0 software. This is a stripped-down version of OpenGL designed specifically for Embedded Systems (ES). It's the graphics standard used on most mobile phones; however, desktop machines use full OpenGL, so most 3D software for Linux won't run on the Raspberry Pi.

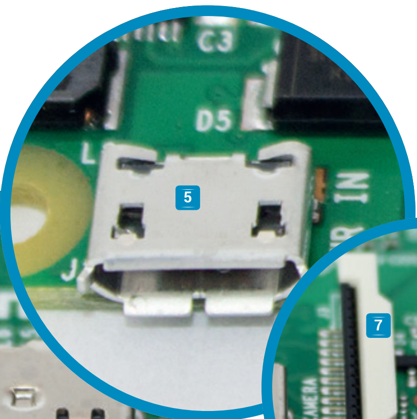
This may be about to change. Eric Anholt, who works on open source graphics at Intel, is working on a restructured driver for the Video Core IV. It should mean a performance improvement, the ability to run 3D applications in windows (rather than just taking over a chunk of the screen) and it should mean full OpenGL applications can run. Potentially, this means much more software should be available for the Pi, but we will have to wait to find out what the performance is like. Eric gave an excellent overview of his work at Linux.conf.au, and you can watch it on YouTube at **https://www.youtube.com/watch?v=EXDeketJNdk**.

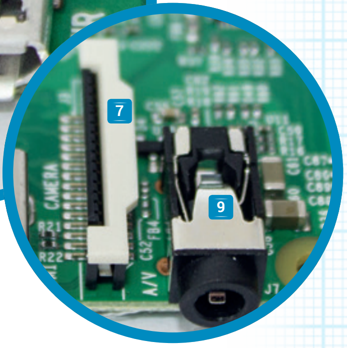
Raspberry Pi 2 model B in detail

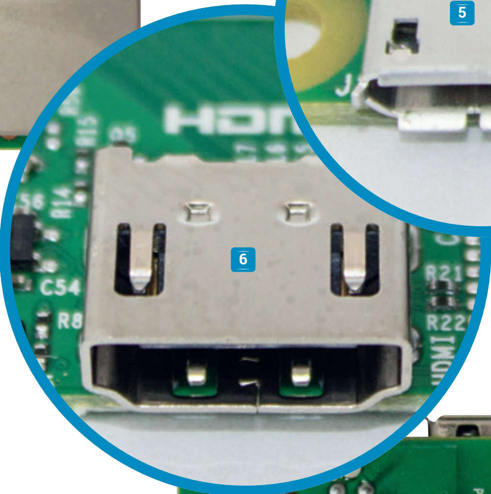
The new components, and what it means for you.



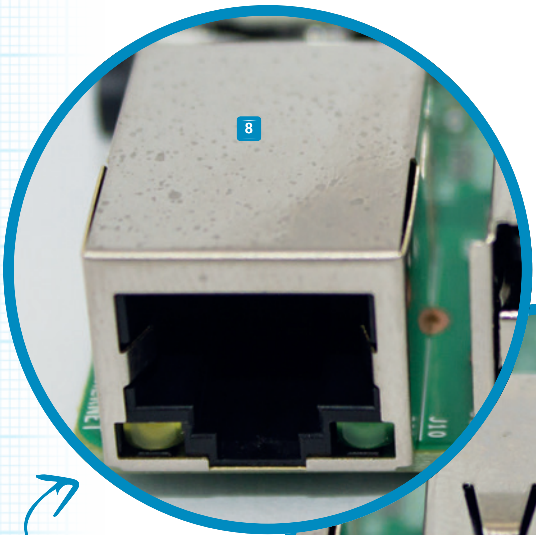
You can now see the wave of the Broadcom logo on the SoC as it's no longer obscured by the memory. [1]

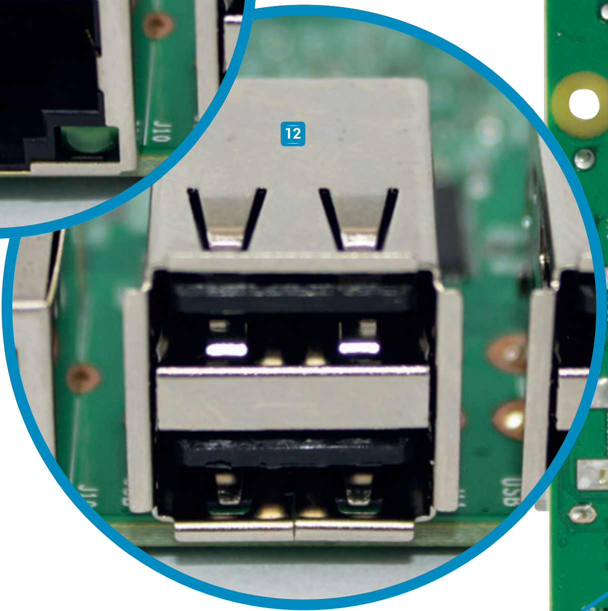




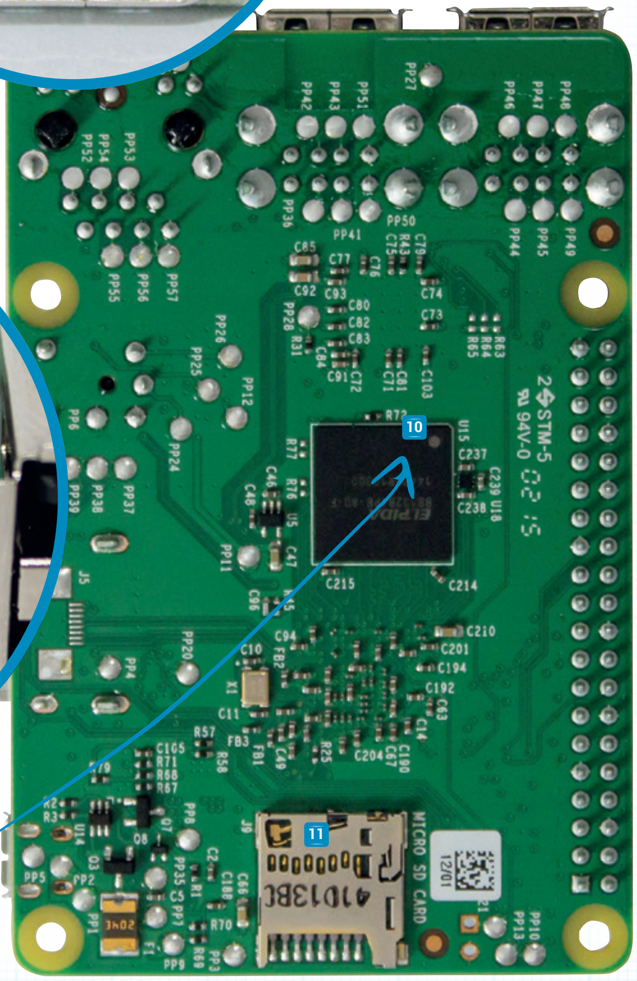


Version 2 looks almost identical to version 1 model B+.





The USB and networking setup remain unchanged, so the speed for networking and external storage haven't increased significantly.



The underside of the new Pi reveals the new, higher-speed RAM. [10]

1 Quad-core ARM Coretex A7 processor The shining jewel in the Pi version 2 is the new System on a Chip (SoC). This consists of four ARM cores, each of which is by itself more powerful than the single processor in the previous version. The impact of this can be seen on the benchmarks on the next page.

2 Video Core IV GPU The graphics processor is the same as on the previous incarnation of the Pi, so you can still expect full HD video playback and smooth OpenGL ES 2.0 graphics.

3 Display connector Although there is no display yet released for the Raspberry Pi that can use the display connector, the Foundation tell us that they're just making the final adjustments to the design and it should be released soon. In the meantime, you'll have to make do with HDMI and RCA.

4 GPIO There are 26 usable input or outputs on the 40-pin GPIO header, which include UART, I2C and SPI buses. This is the same layout as the model B+, and is backwards-compatible with all Raspberry Pis made since September 2012. The very first models had a slightly different pin numbering, so if you're upgrading from an early Pi you'll need to make sure that any expansions you have are compatible with revision 2.0 (almost all are).

5 Switching regulator The Raspberry Pi version 2 retains the same switching regulator introduced in the model B+. This offered a significant improvement on the linear regulator in the original version. However, the increased power of the processors on version 2 means that power consumption is increased by about 1 watt (or 0.2A at 5V). This means that it's more important to have a good power supply with the Version 2 than it is for the B+.

6 HDMI The main video output can also be used for digital audio. The HDMI bus is two-way and can be used to send information from the display to the Pi through the Consumer Electronics Control (CEC) protocol. This is most commonly used in media centres to send instructions from a TV remote control to the Pi.

7 Camera connector This still connects with the normal and low-light cameras from the Raspberry Pi Foundation. The increased processing power of the new model means that advanced picture processing will run faster.

“The memory has come off the SoC, been beefed up to 1GB and increased in speed.”

8 Networking The one slight disappointment for us was the fact that the USB and networking still share the same connection to the CPU. This limits the speed, especially when several USB ports are in use at the same time as the network (such as when you're using a USB hard drive).

9 Audio and analogue video out The analogue outputs run just as they did on earlier versions, and we haven't found any problems with them.

10 Memory The memory has come off the SoC, been beefed up to 1GB and increased in speed. All this means the new Pi is far more responsive when multi-tasking. Web browsing, in particular, is noticeably better on the new version.

11 Micro SD If you still have a version 1 model B (before B+) then you'll need to switch from an SD card to a micro SD card, but otherwise you shouldn't have a problem. The latest version of Raspbian runs on both new and old RasPis, so you can still share storage between different versions.

12 USB There are still four USB ports, and the higher power draw of the new SoC could lead to problems if you have several high-power USB peripherals and a weak power supply. This should be roughly comparable to the situation with the model B before the B+. A powered USB hub will enable you to use more devices.

Understanding ARM cores Overloading on numbers and letters

There are many different ARM processors on the market, and they each have different performance characteristics. The first Raspberry Pi features an ARMv6 processor (also known – somewhat confusingly – as ARM11), while the newer version features ARMv7. These are different instruction sets. ARMv7 is backwards-compatible with v6, so you should be able to run software compiled for the older processor on the newer one. However, there are additional features in ARMv7 that software can take advantage of to run faster .

Most ARM processors available today are ARMv7, but there are still significant differences between different chips, even when they understand the same instructions. Almost all System-on-Chips (SoCs) that run Linux use ARM Cortex A cores (the A stands for application) – these are all ARMv7 or higher. Within this designation the most common cores for small Linux Computers are the Cortex A5 (such as the Odroid C1), Cortex A7 (such as the Raspberry Pi 2) or Cortex A9 (such as the Udoo).

A little confusingly, a company called AllWinner makes SoCs that are also numbered with A's. For example, the AllWinner A20 is quite popular with small Linux computers. However, in this case, the A20 is the name of the chip, not the core, and the AllWinner A20 actually uses a Cortex A7 processor.

In general, as the numbers on Cortex A series processors go up, the processors become more powerful. In this context, more powerful means capable of doing more in a single clock cycle. So, for example, a Cortex A7 processor would perform most tasks about 20% faster than a Cortex A5 processor at the same clock speed. Cortex A9s are also noticeably faster than A7s.

It isn't always performance that differs between differently numbered Cortex A series chips. Sometimes it's that one core has different features (such as hardware virtualisation) than others.

When looking at the specifications of an ARM SoC, it's important to understand that even though Cortex chips are all ARMv7, they don't all have the same performance cycle-for-cycle or core-for-core.

There is a counterpoint to the performance difference, and that's that the lower-performance cores also tend to use less power, so it sometimes makes sense to use a lower-powered chip if you're running off a battery.

To make matters even more complex, some newer SoC's feature more than one type of core. For example, the AllWinner A80 SoC features eight cores. Four of them are Cortex A15s and four are Cortex A7s. This combination of core types is know as the ‘big.LITTLE’ configuration, and allows operating system to select the lower-powered Cortex A7 cores to run more energy efficiently when the load is low, then dynamically switch to the more powerful Cortex A15 cores to run faster when there's a higher load.

You may also come across ARM Cortex M series chips. These are for microcontrollers, so they're not designed to run Linux (or any other OS). For example, the Arduino Due has a Cortex M3.

Thanks to their low power draw, ARM chips are found in many smartphones.

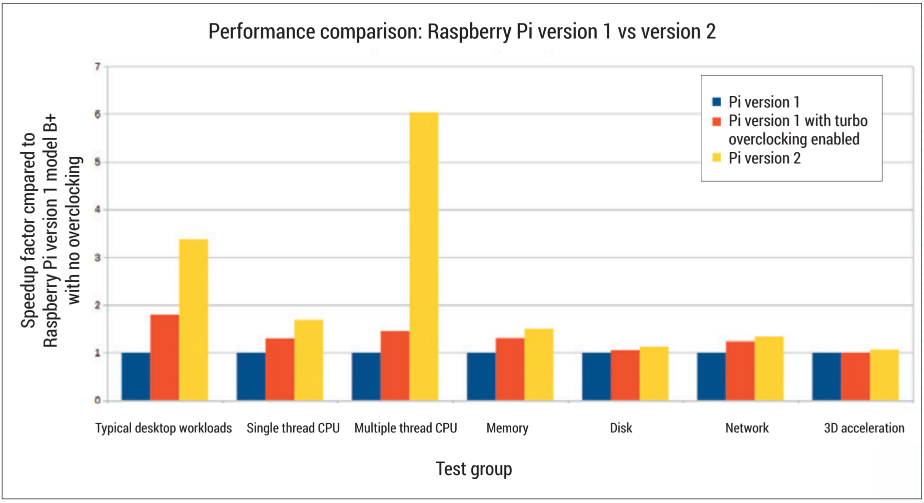
Benchmarking

Just how much faster is the new Pi?

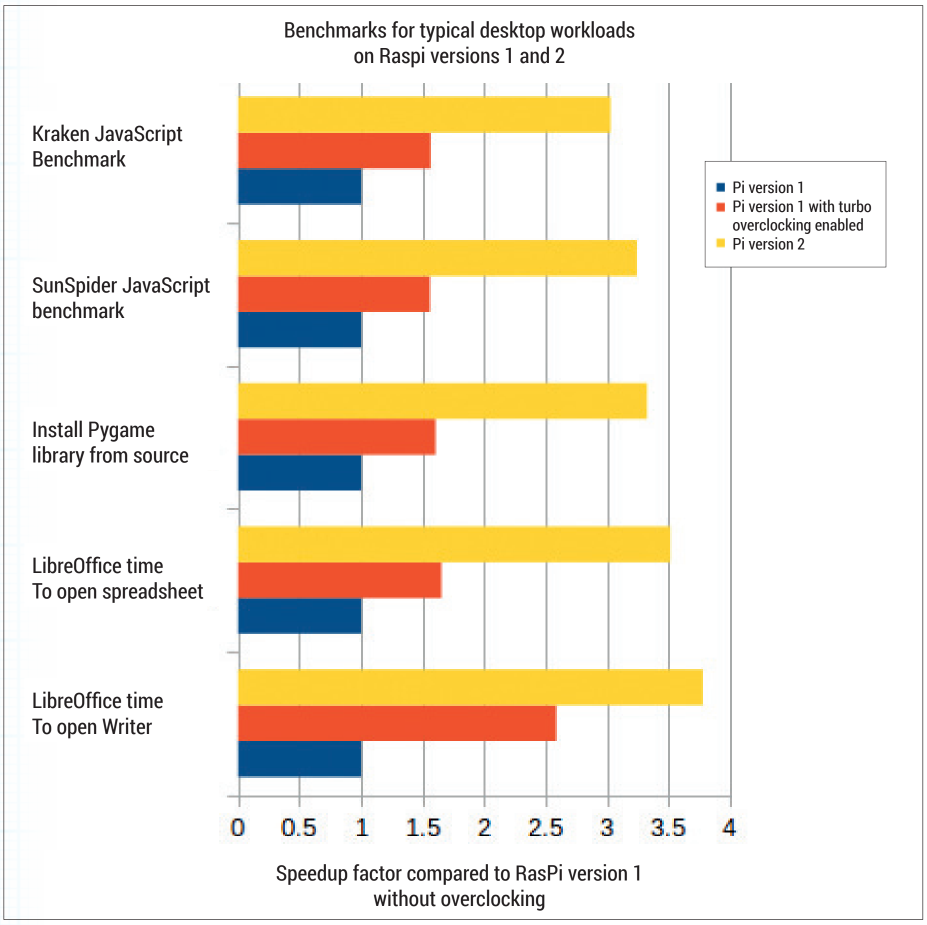
Moore's law (as interpreted by David House) states that the power of computer processors should double every 18 months. If that's true, then the new Raspberry Pi should be four times more powerful than the original, which was released three years earlier. To find out what the actual difference is, we ran a wide range of benchmarks to stress the Pi in different ways. All the tests were carried out at the default clock speed on both versions of the Raspberry Pi (with different levels of overclocking on version 1).

Our battery of CPU tests showed that the combined processing cores on the Pi version 2 can crunch data about 6 times the speed of the single core on version 1. These tests were performed using highly parallelisable tasks such as zipping and unzipping files using the bzip2 algorithm (7.29 times faster) and calculating cryptographic hashes (an average of 5.54 times faster). All these tasks are efficient at splitting the load across the four cores in version 2.

Not all computing tasks are parallelisable like this, and often, you only end up using a single core on a multi-core processor for a particular task. We also ran some single-threaded CPU tests from the **hardinfo** tool that stressed just a single core of the Pi version 2. However, the other cores were running, so background tasks could take place on other cores. These tests weren't necessary a perfect comparison of the processing power of the cores, but a test to see how much of a speedup you should expect to see on single-threaded tasks. The results varied from a speed up of 1.87 (Fast Fourier transforms) to 1.51 times (Blowfish encryption). The average speed improvement was 1.69 times.



The speedup of the extra processing power isn't matched in other areas, but normal usage should see a significant improvement.



Regular workloads tax almost all parts of the system from the CPU to memory to storage, and they do it in different ways.

Not just the CPU

Of course, there's much more to a computer's speed than just raw CPU power. The new Pi also has different memory, which is clocked slightly faster. We were able to write data to memory 1.50 times faster on the new model. One thing these tests don't show is that the new Pi has twice the amount of RAM as the old version. This difference hasn't show itself in any of these benchmarks, but can have a dramatic impact on RAM-heavy tasks, particularly if you're running several tasks as once, like web browsing with multiple tabs open.

Both models use micro SD cards for storage, and a similar setup. Using identical SD cards, we found a modest 1.12 times speedup on the new Pi.

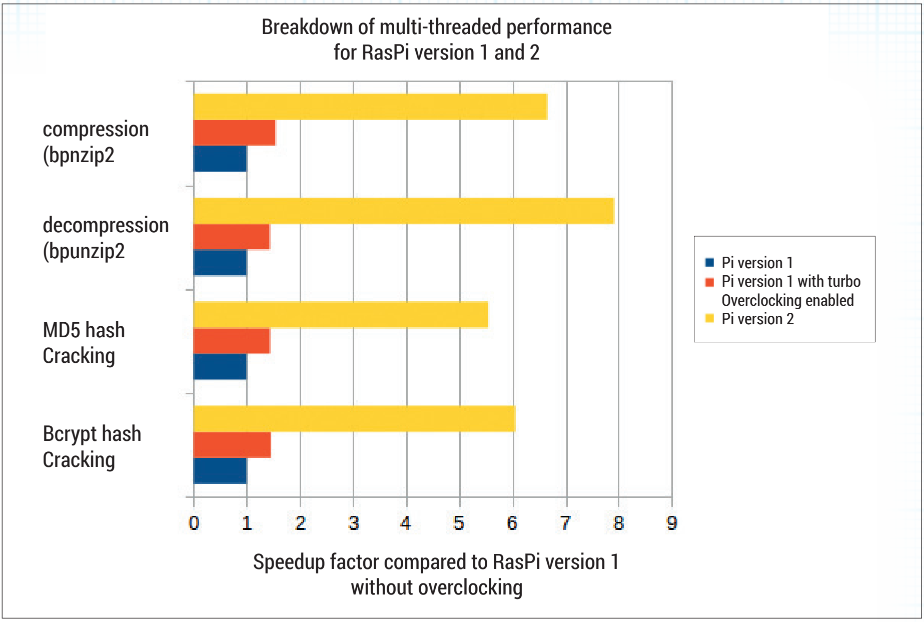
The new Pi has the same network interface using the USB bus as the old Pi.

Here we ran two tests: one to see how the new Pi handled network throughput (using **wget**), and another that tested how the Pi coped with an encrypted download (via **scp**). In the unencrypted test, the new Pi performed about 14% faster than the old Pi. There was relatively low CPU load, so the more powerful SoC didn't have much advantage. However, in the encrypted transfer, the older processor struggled to keep up, and the new Pi performed 50% faster in the encrypted transfer (**scp**) test.

The new Pi has exactly the same VideoCore IV graphics processor as the old Pi, so we didn't expect to see any difference in performance here. We ran a range of tests with a steadily increasing number of objects in the 3D scene, and with varying levels of texture. Across them all, we saw a fairly consistent 6% improvement in speed, which we suspect is due to the CPU side of the benchmark running faster. In other words, we wouldn't expect a noticeable increase in performance using accelerated graphics.

It's telling to see how the two versions compare on benchmarks that target specific subsystems, but most computing tasks run across many of the different areas. They have some parallelisable aspects, and some aspects that aren't. The memory, CPU, and disk speeds all factor into the speed as well. We put together some benchmarks to try to capture the overall speed of the Pi under normal use. This included tests on *LibreOffice*, web browsing and installing software. Despite very different workloads, they all reported a fairly similar speed up with the new Pi, about a factor of three. The biggest improvement was opening *LibreOffice* (3.78 times faster), while the smallest was the Kraken JavaScript benchmark (3.03 times improvement). Opening a complex spreadsheet in *LibreOffice Calc* came in 3.51 times faster.

"The new Pi dramatically outperformed the overclocked old Pi in multithreaded benchmarks."



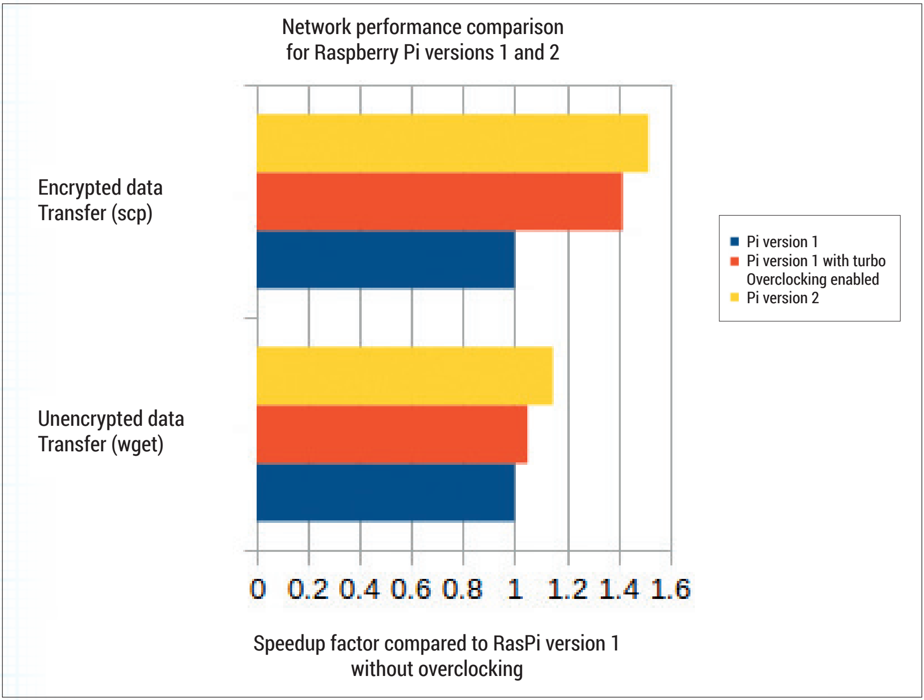
The multithreaded CPU tests were perfect for version 2 to show off its processing power.

When we added them all up and averaged them out, we found that the new Pi is 3.38 times faster than the old Pi under common loads. We're confident that this is accurate for typical workloads, but obviously, different tasks will vary as they will depend on the speeds of the different parts of the Pi in different ways.

The speedup factors were calculated by comparing the new Pi to an old one at the normal level of overclocking. However, it is possible to overclock version 1, so we also ran all the tests with turbo overclocking enabled. In this case, we found that the single-threaded performance was only slightly faster on version 2. However, the new Pi still dramatically outperformed the turbo overclocked Pi in multithreaded benchmarks, and was almost twice as fast at typical desktop tasks.

All of these benchmarks evaluate some form of measurable metric, but they also hide the important but hard-to-measure aspect of how the computer feels to use. The biggest improvement in this area is the responsiveness of the new version. The old single-core machine was prone to locking up and becoming unresponsive under heavy load. This aspect of the Pi is now gone.

What does all this mean? Turn the page to find out how the different performance will affect different types of use.



The speedup in encrypted transfer came as a bit of a surprise to us, but is useful for many setups.

Popular projects

How the new performance profile will change the way you use the Pi.

Gaming

Party like it's 1989.

There are only two common uses for the Raspberry Pi in gaming: *Minecraft* and console emulation. There hasn't been an update to *Minecraft* for the Pi since it was first released, and we've no indication that a new version is likely. Even if a new version did come, the Pi 2 is unlikely to perform significantly better. Perhaps the biggest improvement here is for people writing programs that interact with the *Minecraft* world, as they now have significantly more processing power to use.

Playing old games on the Pi is well supported through distributions like RetroPie. The original Pi handled the emulation of most older consoles fairly well, albeit with occasional jitters. The newer version's processor is only about 50% faster on single-threaded tasks and only marginally faster than a turbo-overclocked Pi 1. Since most emulators don't take advantage of multiple cores, we wouldn't expect the new Pi to really shine here unless some of the multi-core emulators are ported. That said, the new Pi shouldn't experience and problems if background tasks start during the emulation.

```
etest 0.4.10-dev_MSVC
g1> :D ::D :D
ne of day set to 6am
unman> tron do you wan't be freind
ron> yay
ron> sure funman
ron> do u have a house
```

Minecraft now has a more Pythonic Python 3 API for controlling the world from your programs.

Desktop computing

Desktop usage is probably the area where the improved performance of the new Pi will be most obvious. In our general-purpose benchmarks on the previous page, we found a roughly 3.4 times increase in most tasks, which is a huge difference.

The biggest difference, though, isn't in the time it takes to run a single task, but in multi-tasking. Things like **apt-get**ting software don't render the machine unresponsive, and neither does a single JavaScript-heavy web page. The increase in RAM is also important, because it means you can run a lot more software before the system grinds to a halt as it shuffles memory in and out of swap space.

If you're currently using a Pi as a desktop computer, we'd strongly recommend you get a new Pi 2. The performance increase is definitely worth the $35. We can't think of a time in the history of computing where such a small amount of money could have such a big performance impact.

We also expect both versions of the Pi to get a noticeable performance improvement with Wayland, which is currently under development.

Home theatre PC

A DIY smart telly that you control.

The developers of *XBMC* (now known as *Kodi*) have done an excellent job of making sure the Pi version 1 can be used as a home theatre PC (HTPC) to play videos and music. The Pi version 2 uses exactly the same GPU as version 1, so all that good work automatically flows over and the Pi 2 should make an excellent HTPC. Using the same GPU means that you shouldn't expect any significant improvement in the video playback. That said, video playback usually works well on the Pi.

The improvements on the new Pi should be noticeable in other areas. Most media centre interfaces aren't accelerated, so the improved speed of the new Pi should result in a better user experience. Exactly how much better will depend on what media player you're using, and what skin it has.

The Pi 2 has more in common with other ARM boards than the Pi 1 did. This is both down to the instruction set and the memory layout. This means that – in theory at least – it should be easier to build Android for the Pi. There isn't likely to be an official release from the Foundation, but a community project is possible. This would provide another option for HTPC interfaces, and should mean that streaming media services like Netflix would work on the Pi.

5/14. My Shop: Nisha Anil and her panda friends take a look at children's lives in the UK. Nisha, Mish, Moosh and Mogo head across the sea to meet some children in Northern Ireland. [HD] [AD,S]

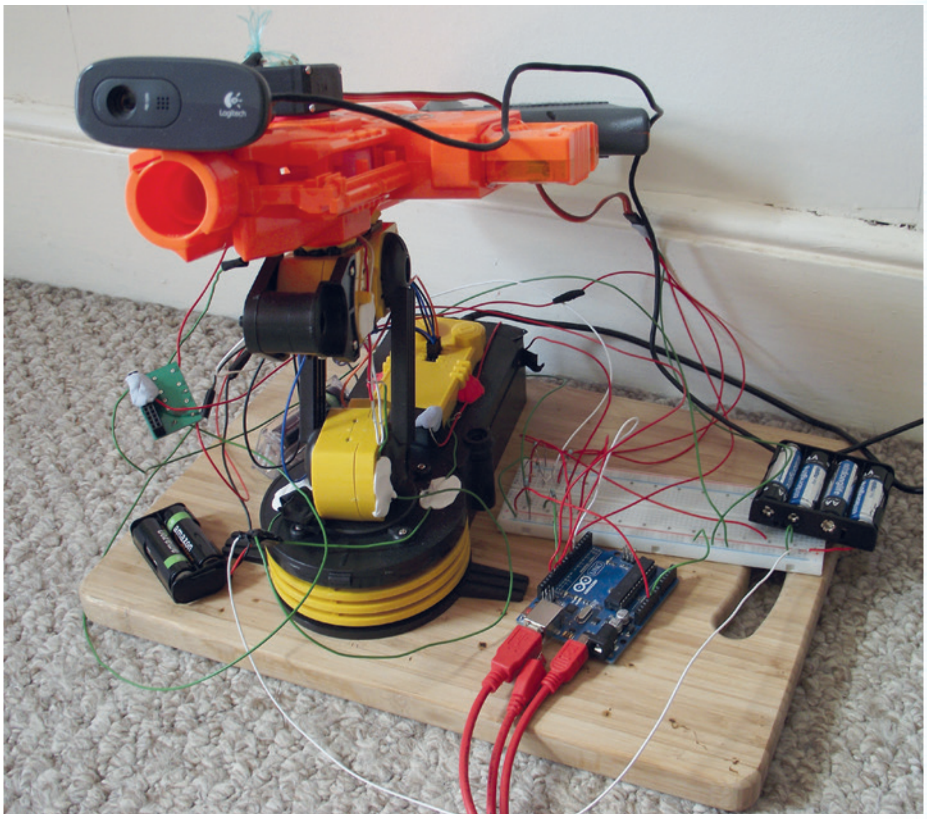
We covered setting up a *Kodi*-based media centre in issue 12.

Robotics

More brains for your mechanical army.

Pure processing power is very rarely a problem in hobbyist robotics. The first version of the Pi easily has enough power to control motors, fetch data from sensors, and handle the camera. The limiting factor with controllers for robots is almost always the connectivity (GPIOs, I2C busses, UARTs, etc). In this regard, version 2 offers no advantages over the older B+. In fact, the higher power usage could actually make it a little worse in some cases. Generally, the A+ is the best board for robotics when power consumption is at stake, and the A+ will continue to use the same SoC as before.

There are, however, a few cases where robots need more processing power. For example, in machine vision. OpenCV, a library written for object recognition, and has some uses in robotics – we used it in our face-tracking Nerf gun in issue 4. It does run on the Raspberry Pi version 1, but only just. You have to limit the resolution, and you still get quite a slow framerate. This means you don't get as accurate recognition as on more powerful computers. Version 2 should significantly improve things here, and make Raspberry Pi robots' vision more powerful.


Now your robot weaponry can have better vison, yet still have the portability of the Raspberry Pi.

Programming and more

The ultimate tool for teaching children to code?

The primary purpose of the Raspberry Pi is to teach computing to school children. The new version makes this task easier. A significant part of this is that general desktop computing runs better.

One task that always frustrated us with the earlier version was installing libraries. Sometimes it felt like we were waiting five or 10 minutes before we could get back to programming, and this was very off-putting for beginners (and quite a problem when teaching a class). The task also locked up the only core, so we couldn't do anything while the libraries were installing. Version 2 ran 3.33 times faster than version 1 when compiling and installing the *Pygame* module. Obviously when coding in a compiled language, you should expect similar speed ups in general compilation runs.

Scratch, a graphical language for getting children interested in programming, should also perform significantly better on the new Pi. The new model is all round a better machine for people new to programming and experienced coders. That said, it's still not suitable for heavy programming tasks, and it struggles to run heavyweight IDEs such as *Eclipse*.

Personal server

Then there's the use of the Pi as a personal server. At the simple end, this could be a machine hosting Samba shares; on the other end, it could be a full LAMP stack running a range of web applications.

The new model has the same network and storage setup, so you should only see fairly modest improvements in simple file-sharing, unless you use encryption, in which case you may see a 50%+ improvement.

Most web servers (including *Apache*) are good at splitting load across multiple cores, so if you're using a web front-end, you could expect a more significant speed up. The exact speedup will depend on how you use the server, but the new Pi should be much better at serving more than one request at a time.

The road ahead

We're excited by the new version of the Pi, and you should be too. Part of this excitement, however, comes not from what's new, but from what has been retained from the last version. The Raspberry Pi's video capabilities have always been excellent and video is still excellent in the Pi Version 2. The same goes for the layout of the GPIO pins. Countless hackers have built robots, submarines, weather stations and more with Raspberry Pi, and the GPIO pins provide the interface between the brains of the computer and whatever sensors, motors and servos are used. If this changed it would damage the work that third parties have put into building the Raspberry Pi community, so we're pleased to see it stay the same.

What has changed though is the feel of using the Pi. Plug it into a keyboard and monitor and it feels so much more like a grown-up PC, and we think that this will make it even more attractive to schools who need cheap hardware to teach kids that computers aren't mysterious black boxes. Add to that the improved capabilities in just about every area and the new Raspberry Pi Version 2 is on to a massive winner. We can't wait to see what's next. LV

Schneier was at FIC 2015 to talk about how companies lose control of security when they outsource infrastructure. (Photos: Laurine Dutot)

A QUICK CHAT WITH BRUCE SCHNEIER

One of the biggest names in security responds to government snooping plans.

Bruce Schneier is one of the best known experts on security and encryption issues. He has written many books on security and cartography over the last two decades, and maintains a well-followed blog at **www.schneier.com**. He coined the term 'security theatre' to describe the ludicrous hoops we have to jump through at airports that don't actually make us safer, but are instead aimed at making us feel safer, and he's the first person we google when we want to find out the real truth behind the latest 'think of the children' security scare. So we were delighted to get the chance to meet him at the FIC 2015 cybersecurity conference in late January (see page 18), to ask about Edward Snowden, privacy and recent government plans to spy even further into our communications.

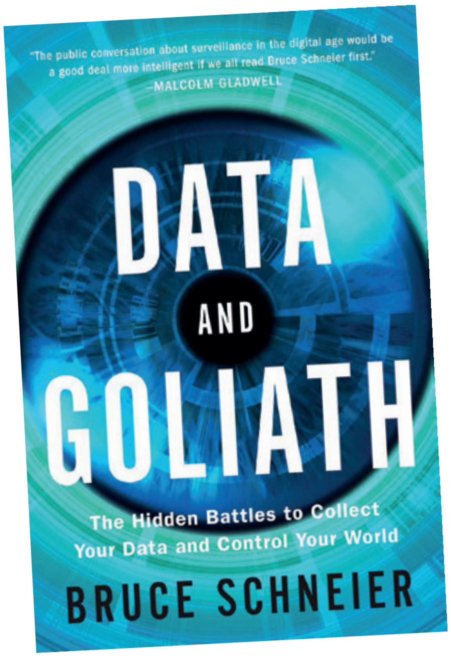
LV You've no doubt heard that Prime Minister David Cameron in the UK and now President Obama have said that they want access to all encrypted communication on the internet. What was your reaction when you read that?

Bruce Schneier: It's completely idiotic! We've heard about this idea since the mid 90s – it's not new at all.

Is it workable in any way?
BS: Of course not. Only someone who is not a technologist would say such a thing. The problem is, I can't build a back door that only works for people of certain morals. I just technically can't do that – I can't design a filter that filters for morality. So if the US government can break in, anyone else can break in.

Or someone from the government leaves a laptop on a train, containing the encryption keys. It's happened before...
BS: Right. So it's unworkable and impossible, but we heard FBI director James Comey talk about this in November. But this is like former FBI director Louis Freeh in the 90s, this is like the Crypto Wars*, and it's back again. Not only will it make us all insecure, it won't even do what the government wants.

But politicians will always use arguments like "think of the children", and people will keep voting for them...
BS: Indeed they will, because fear sells. In the 90s I talked about the "four horsemen of the internet apocalypse" and they were: terrorists, drug dealers, child pornographers and kidnappers. All this fear. So we have to win this. We can't say that we will all be insecure forever on the internet. That's just crazy.


"The public conversation about surveillance in the digital age would be a good deal more intelligent if we all read Bruce Schneier first."
—MALCOLM GLADWELL
DATA AND GOLIATH
The Hidden Battles to Collect Your Data and Control Your World
BRUCE SCHNEIER

Schneier's next book is due out in April – stay tuned for a review in a future issue.

You would expect, after the Snowden revelations, that everyone would be fighting more for privacy on the net.
BS: You know, some people are – the IETF [Internet Engineering Task Force] is. Certainly Google is and Microsoft is. What Google is doing about the US data in Ireland – they are encrypting more of their stuff. I think the Snowden leaks caused the company to step back and say "we shouldn't cooperate here".

You don't think it's just bluster from companies trying to keep hold of their customers?
BS: I think it's half bluster and half real. But there is progress there – we are seeing that companies are fighting.

You worked with Glenn Greenwald going through the Snowden documents. What surprised you the most?
BS: The thing that was most surprising was: there were no surprises. The amazing thing about the NSA is that they're not made of magic. You'd think, with their budget and the amount of personnel they have working for them, they'd have some magic in there. But they kind of don't. We learnt that they don't have quantum computers doing amazing things – they're just better funded than the typical hacker.

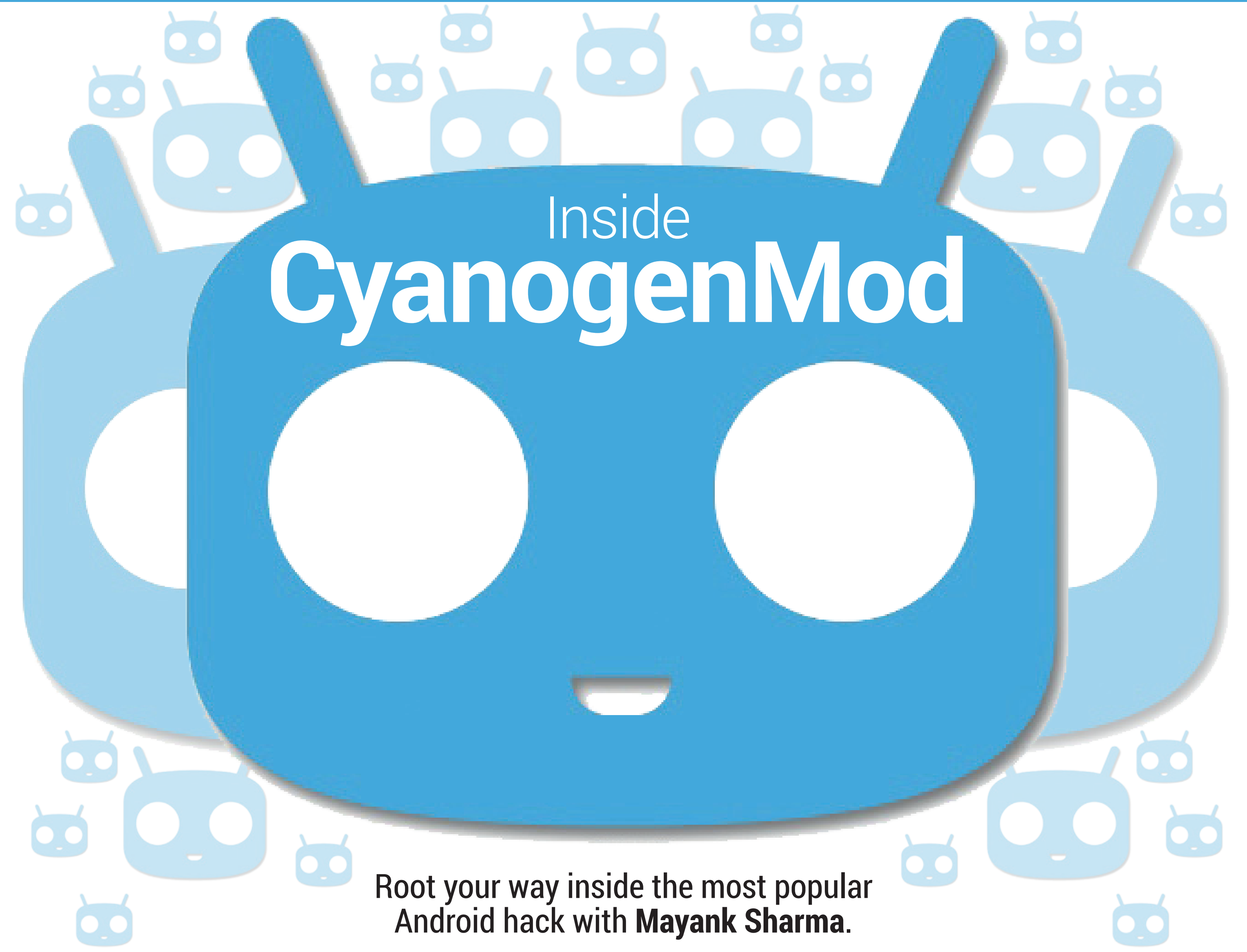
Like: wow, they've broken some encryption system that looked almost impossible to break?
BS: Right. There was no "oh my god, they can do that?" Their stuff is just souped-up hacker tools. I thought there'd be something big, like a break on AES [Advanced Encryption Standard].

Do you think much has changed in the meantime? There's a drive towards HTTPS everywhere, for instance.
BS: I think there has been. It's around the edges, but there have been changes. WhatsApp is encrypted for instance – that's 700 million people encrypting. That's amazing.

Some services like Google Mail rely on being able to see content, to sell advertising...
BS: I think Google could say: "You know, there's not a lot of money in email advertising, so let's just encrypt it". They could do different sorts of advertising – just not content-based.

Anyway, there's a lot we can all do to make things better. Every time you use encryption with nothing to hide, someone else who needs encryption to stay alive benefits, because he's hiding in a bigger pool of encrypted data. Just like every time you use *Tor*, you make it better for someone else who needs to use *Tor*. There's safety in numbers.

*** LV note: Louis Freeh was Director of the FBI from 1993 to 2001, and described encryption "one of the most difficult problems for law enforcement as the next century approaches". Crypto Wars refers to the US government's attempts to limit general access to encryption technologies that it couldn't break itself.**



Inside CyanogenMod

Root your way inside the most popular Android hack with **Mayank Sharma**.

The CyanogenMod project is the quintessential open source success story. It's got open source code, an individual hacker burning the midnight oil, and community endorsement that led the project to the big leagues. From a forum board to the executive board, the CyanogenMod project has come a long way riding on the shoulder of its community of users and developers.

With over 12 million active users as of June 2014, CyanogenMod is the most popular free and open source aftermarket firmware for Android smartphones and tablets. The tricked-out version of Android replaces the stock OS that ships with your device and essentially gives you more control over it.

CyanogenMod is built upon the open source release of Google's Android, known as the Android Open Source Project (AOSP). In 2008, developers found a way to gain root access to the HTC Dream G1, which shipped with Android 1.0. This allowed those with the technical know-how to replace the stock Android with a customised flavour of their own built atop the AOSP. This development spawned a community of modders who showcased their mods at places like the XDA Developers forum. Steve Kondik was one such developer. He called himself Cyanogen online, and therefore named his creation CyanogenMod.

CyanogenMod quickly grew in popularity and started to attract modifications and improvements from other developers as well. Kondik rallied a bunch of active contributors into the CyanogenMod Core Team to accept code based on the feedback from their community of users.

"CyanogenMod is built on the open source release of Google's Android operating system."

Anatomy of a hack

CyanogenMod officially supports over 200 devices and there are experimental builds for a lot more, which is an incredible achievement once you take into account the elaborate process it takes to get

CyanogenMod to run on a device. Add to this the fact that the developers have to create a different version of CyanogenMod for each new Android release, and that each device has different hardware components, and you have some idea of the scale of the job.

To keep up with the proliferation of Android devices, the CyanogenMod team is divided into various device maintainers, who manage the code for different devices and make sure the device they're in charge of runs CyanogenMod properly. One such device manager is Chirayu Desai, who is currently in the last year of high school. The young padawan is in-charge of "almost all" supported Sony devices. However, he got his start with another device. Recounting the start of his association with the project, which dates back to before Android 4.0 Ice Cream Sandwich (ICS) was released in 2011, Desai says "I had a Samsung Galaxy Tab, which was one of the first Android tablets, and it shipped with Android 2.3, which was not something you would like using on a tablet."

Desai almost immediately swapped out the stock Android and replaced it with CyanogenMod: "When ICS was announced, I collaborated with some other developers, and seeing that it was working on two other very similar devices (the Nexus S and Galaxy S) we wanted to get it working on the Tab too." That's when he started porting the current release of CyanogenMod: "When it was in a decent state, I contacted the CyanogenMod developers and was able to get it officially supported."

We asked Desai to outline the process a developer needs to go through to port CyanogenMod to a device: "One of the first things that you need to find out is what hardware the device has, and what information can you get about that – what System-on-Chip (SoC), which sensors, etc. Then you need to see what kind of source code is available from the concerned OEMs, and for devices similar to the one you are working on, which is to say, other devices that have the same or a very similar SoC. Then you put everything together, adapt it for the Android version you're porting, and for standard interfaces as needed." Since the project supports a lot of devices and many different hardware platforms and shares code between them, Desai adds that the developers also "have to keep in mind that if our device needs a 'hack' somewhere it should not affect the others. Every release breaks things in ways that can be fixed, worked around, or need a hack – it varies a lot by device."

CyanogenMod's co-founder and CTO Steve Kondik (left) and CEO Kirt McMaster.

The empire strikes back

CyanogenMod owes its existence to the open source vanilla Android, but Google adds a different flavour to the mix. CyanogenMod initially bundled Google's proprietary apps, Gmail, Calendar, and others along with its software, but Google intervened with a cease-and-desist letter and prevented the project from distributing its mod until the project stopped including Google's apps in CyanogenMod.

The two also had an exchange over CyanogenMod's installer. The project pulled it voluntarily from the Play Store after Google communicated that it was at odds with its terms, since it encouraged users to void their warranty. Despite these run-ins, many mobile

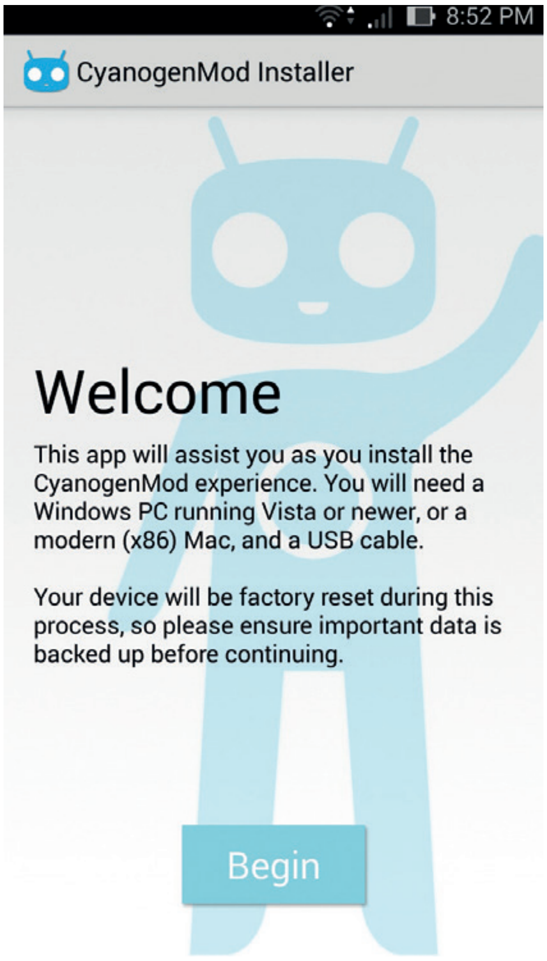
Kondik rallying his troops, startup style, in Cyanogen Inc's Seattle office.

Top The Oppo N1 has a swivelling camera and includes CyanogenMod's advanced camera app. **Top right** The OnePlusOne includes CyanogenMod's Trebuchet Launcher as the default home screen.

commentators have said that Google has borrowed several features from its popular replacement.

"Google's methodology seems to be re-imagining community-driven features, much like we re-imagine their Android updates", said Steve Kondik in an email exchange with us. "One of the earliest examples of this was incognito mode on the Android browser, prior to Chrome for Android. The implementations were different, but the idea – privacy on mobile browsing – was the same."

Another feature Kondik points out is the Android Quick Shortcuts: "Not only did we have a working implementation before Google, called Notification Power Widgets, but even after Google implemented their own system in Android Jellybean, a single tap in their implementation would launch the Wi-Fi settings (for example), whereas in CyanogenMod, that same tap would immediately toggle Wi-Fi on or off. As we look at Google's implementation in Android Lollipop, you can see that they've adopted the same behavioural change." In the end, he sums up, "it's always nice to see this cycle of changes between community projects and Google alike."


8:52 PM

CyanogenMod Installer

Welcome

This app will assist you as you install the CyanogenMod experience. You will need a Windows PC running Vista or newer, or a modern (x86) Mac, and a USB cable.

Your device will be factory reset during this process, so please ensure important data is backed up before continuing.

Feeling lazy? Borrow a Windows machine and save yourself a considerable amount of effort by installing CyanogenMod with the graphical installer.

An Inc.ling

One of the core issues that the project's leadership has been working to address is to simplify the process of getting their mod onto a device. Given that the process of replacing the firmware on a device requires considerable technical expertise, the project is doing well to have won such a large userbase. Also, the CyanogenMod project has always been vocal about its ambition of being more than just a software mod. In a *Wired* article in 2011, Chris Soyers, who has been associated with the project for a long time, said that one of the project's biggest dreams is to see a phone ship with CyanogenMod on it.

In fact, these were the two main catalysts that propelled Kondik to act on his plans to commercialise CyanogenMod. Cyanogen Inc. was announced in September 2013 and managed to raise $7 million (about £4.6 million) from venture capitalists and another $23 million (about £15 million) later that year. Kondik says that having a company with full-time staff and dedicated resources enables them to make bets on larger projects, particularly those that would take too high of an investment from the community alone, or would be a challenge to coordinate.

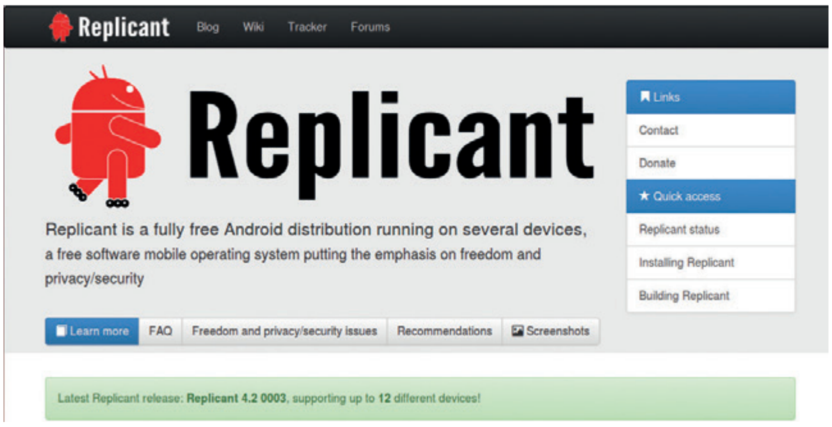
"The enhanced theming capabilities that we bought last year and the Encrypted Text Messaging (partnering with Open Whisper Systems) the year before are both examples of these larger long-term and impactful projects," he illustrates. Furthermore, he adds that since the code is open source, these new additions are not only beneficial to the company but to the community as well.

The CyanogenMod advantage

Why would millions of users go through the pain of replacing the stock OS on their Android device with CyanogenMod, at the risk of voiding the warranty? What makes CyanogenMod so special?

While Android allows access to things that other vendors (Apple) prevent, such as installing apps from unofficial sources, CyanogenMod takes this control to a higher level. For starters, CyanogenMod gives you root access to your device, which enables you to remove any app, including systems apps, from the device. The mod is also known for its customisation and privacy-enhancing features.

CyanogenMod's theme engine lets you modify the look and feel of the entire OS. It also comes with privacy features such as the Privacy Guard, with which you can control the information your device shares with individual apps. The mod also includes a global blacklist that lets you block messages and calls from unwanted contacts. Furthermore, if you get yourself a CyanogenMod account you'll have the ability to locate and wipe your device remotely should it be stolen. There's also a hardware assisted disk encryption in the newer version.



Replicant is a fully free Android distribution running on several devices, a free software mobile operating system putting the emphasis on freedom and privacy/security

Latest Replicant release: Replicant 4.2 0003, supporting up to 12 different devices!

The Replicant OS based on CyanogenMod replaces all proprietary Android components with their free software alternatives.

Having a company also allows them to impress upon device vendors their approach to total access to the hardware "without voiding the warranty and preventing support for hardware issues," explains Kondik, adding that the community will "not only benefit from the tighter integration that such partnerships naturally bring, but also gives them peace of mind and a larger value when they begin tinkering with their devices."

Soon after the company's formation there were concerns among the CyanogenMod community following the announcement that it plans to adopt closed-source licensing for some of the future developments. Kondik responded to the concerns and made it clear that the company had no plans to close-source any of the existing stuff. He also reaffirmed their commitment to the community and explain how the dual licensing model will offer "a stronger degree of protection for contributors" and also gives the company a competitive edge.

One such closed-sourced product is the CyanogenMod installer. It's an easy-to-use app that lets you install CyanogenMod onto your device in a couple of simple steps. Kondik explains why the installer is currently only available for Windows and Mac OS X: "When we were building the installer we took a look at our user demographic from our IRC support, user forums and community website. By and large those who were hitting obstacles during a manual command-line install came from the Windows and Mac families – and as such, we focussed our first releases of the installer to those platforms." That said, Kondik adds that while a Linux version of the installer is on their roadmap, "given the task of deciding where to use our resources, the other two platforms carry priority."

Since the formation of the for-profit enterprise, instead of one product, the team now works on two release branches – CyanogenMod and Cyanogen OS. CyanogenMod is the open-source, community-driven Android OS that's supported by Cyanogen Inc. On the other hand, Cyanogen OS is the commercially-distributed operating system that comes with proprietary features, services and enhancements. Cyanogen OS is what you get when you buy a device that ships with CyanogenMod pre-installed.

"Cyanogen OS is what you get when you buy a device that ships with CyanogenMod pre-installed."

Going places

As CyanogenMod enters its second year of existence, the company has inked deals with a bunch of device vendors who are already shipping smartphones preinstalled with Cyanogen OS all around the world. Their latest handset, the Micromax Yureka Yu, has hit a chord in price-conscious markets like India, and Google is also aggressively targeting India with its low-cost Android One devices, and interestingly these devices are now officially supported by the CyanogenMod project.

So can we expect a Cyanogen-branded phone in the near future? "We are very aware of our strengths, and currently that is in the scope of software and web services around them," explains Kondik adding that "the hardware manufacturing market is a game of volumes and margins, and not something we'd jump into without a team dedicated to figuring out all the intricacies and hurdles around it."

For now, the team plans to stick to software and relegate the hardware bits to experienced hardware partners. "We want to see CyanogenMod, and our commercial offering Cyanogen OS, perceived as a premium OS, a must-have, instead of the underdog perception that followed coming from a community project. [We're] continuing to push forward what makes CyanogenMod so great – that fearlessness to innovate and improve upon the Android experience".

WHERE SHOULD OUR PROFITS GO?

This magazine donates 50% of its profits back to the community. But who should get the money? You decide...

Back at the end of 2013, when we decided to create Linux Voice, we had many discussions about what form the magazine should take. We all had various ideas about the content, the style and the design, but one thing was absolutely clear: we wanted to give something back. We depend on Linux and Free Software, and because many FOSS projects have little (or zero) outside funding, we wanted to help them somehow. We all use Linux and FOSS every day, so it was important to us that we get involved in the community that serves us and you, the readers.

We had several ideas about how do this, and eventually we settled on this: we will give 50% of our year-end profits back to the software, services and communities that help us. In this way we can really help some projects, and still keep some money to grow our little company and hopefully win over more people to Linux as the years go by. As this is our first year, where we've invested a lot in establishing and growing the magazine in newsagents around the world, we don't have a huge pile of cash to throw around. But after juggling the maths, we've worked out that we can give £3,000 back.

It's also important to us that you, the Linux Voice readers and subscribers, have a say in where this money should go. You've helped to get us off the ground and build us up to a successful magazine, and we're endlessly grateful to the awesome community that has talked about us on Twitter, recommended Linux Voice to friends, and given us valuable feedback to improve the magazine over the last 12 months.

In late January we asked our website visitors to come up with ideas for a shortlist of projects and services that deserve our profits (**http://tinyurl.com/kt9hgnk**). Over the next few pages we'll explore the most popular suggestions, and then, at the end of the article, we'll invite you to cast your vote. So first of all, let's see who could really benefit from our money...

Scribus

For a 100% FOSS-built magazine.

In making Linux Voice, we use Free Software extensively: Debian, Arch, Xubuntu, Fedora, *Nano*, *Vim*, *LibreOffice*, *Gimp*, *Inkscape* and many other tools. There's one component in our workflow that's not FOSS, though, and it's *InDesign*. Yes, we use Adobe's proprietary tool for layout, and we know that many readers would love us to jettison it. It's not a simple job, though: *InDesign* (for all its proprietary badness) is a hugely capable program, and most designers are innately familiar with it. We (the editorial team) are geeks focusing on tweaking and hacking our installations, so when we set up Linux Voice, we needed a layout solution quickly. *InDesign* was the sensible choice at the time.

But! Since then, we've started to investigate *Scribus* (**www.scribus.net**) more and more. We know that *Scribus* is also a very versatile desktop publishing program, and has been used commercially in newsletters and other publications. It's not a change we can make overnight, but our art editor is learning its tricks and foibles, and we may be able to make the transition. Many commenters on our website suggested that we fund *Scribus* development for features and fixes we need – but how doable is it?

Money talks

Fortunately, it looks very doable. We talked to Craig Bradney, one of the lead developers of the program, who said: "We'd certainly appreciate some extra funds to help us be able to meet up more and develop and release faster etc, so if that were possible, then great." He suggested that during our *Scribus* testing, we set up a bug tracker account and use it to report problems we encounter, or features that we need in our workflow.

If our profits could also be used for a *Scribus* developer meetup, that would also be great. Because most FOSS development is done over the net, it doesn't seem like face-to-face meetings would make a big difference in terms of code output. But as we've seen from countless meetups and hackathons around the globe, when a bunch of coders get together in person, they spur each other on and can get a huge amount of creativity out of just a few days.

Scribus is the flagship FOSS DTP software – we'd love to be able to use it to make the mag.

Tor exit nodes

Free access to information for all.

Linux Voice reader Félim Whiteley had an unusual idea: "Rent a beefy virtual machine for a year to run some extra *Tor* exit nodes. You are in a much stronger position (freedom of press maybe!) than a home connection to fight any frivolous lawsuits if it were abused." He has a great point, and the more exit nodes there are in existence, the more robust the *Tor* network becomes. *Tor* provides an extra level of security (and partial anonymity) when browsing the web, by routing your HTTP(S) traffic through a series of machines on the *Tor* network. In this way, you can access sites without telling them your IP address.

So, you connect to one *Tor* server (node), which then passes on your web request to another, which then passes it on to another, and so forth. Your request goes through several different nodes before leaving an "exit" node and going out onto the internet. You can't easily be traced back through the sequence of *Tor* nodes, but you still have to be careful. Many people operate the in-between *Tor* nodes, as it's perfectly safe to do so, but operating a web-facing exit node can be dangerous. If you have a *Tor* exit node running in your house, and someone is using it to access illegal material, it could look like you are accessing that material yourself.

However, it would be possible for us to rent a virtual server and host an exit node. *Tor* is immensely useful for people living in oppressive regimes, where they could get into trouble for simply looking at resources like Wikipedia. It also helps whistleblowers spread information by giving them a certain level of anonymity. We could run an exit node, or donate to **www.torservers.net**, which looks after many exit nodes around the world.

Tor, "The Onion Router", is a vital weapon in the battle for privacy and security online.

Software Freedom Conservancy

Promoting, developing and defending FOSS projects.

The Software Freedom Conservancy (**www.sfconservancy.org**) isn't as well known as the Free Software Foundation, but it does important work in providing infrastructure and legal support for Free Software projects. It's a non-profit organisation based in New York that looks after a bunch of well-known projects such as *BusyBox*, *Inkscape*, *Git*, *Wine*, Foresight Linux and *PhpMyAdmin*. It's also responsible for identifying GPL violations and coordinating legal responses – such as when the *BusyBox* suite of command line utilities is used in embedded devices, and the vendors don't release the source code (which has happened many times).

The Conservancy also aims to help other non-profit organisations by developing accounting software. This should "help all non-profits (in free software and in other fields) to avoid paying millions of dollars in licensing fees for sub-par accounting software". The Conservancy also has backing from various large companies and organisations such as Google, Samsung, Red Hat, and the Mozilla Foundation.

software freedom conservancy

Donations from big companies and individuals alike help to promote FOSS and defend copyright infringements in the courts.

Gimp

Maybe we can speed up the 3.0 release?

Sometimes it looks like *Gimp* development is in the doldrums. We've been waiting years for a full implementation of GEGL (the Generic Graphics Library), which will bring in support for higher bit-depth images than the current release, along with non-destructive editing operations. *Gimp 2.10* is supposed to have mostly complete GEGL support, but when will it arrive? The last major release, *Gimp 2.8*, came out almost three years ago. And the project's roadmap doesn't help much either – it hasn't been updated for nearly two years.

Then there are plans for *Gimp 3.0*, which will be ported to *GTK 3*. This will help it to integrate well with Gnome 3 and Cinnamon desktops, and make the program look and work far better on high resolution displays, which are increasingly popular among graphics artists.

A portion of our profits could help *Gimp* on the road to its 2.8 and 3.0 releases. The software accepts donations via the Gnome Foundation, and a chunk of such money goes towards the annual Libre Graphics Meeting event, which brings developers together to share ideas and hack on code.

It's been a long while since a major *Gimp* release – maybe we can speed up development!

Open Rights Group

Promoting rights in the digital age.

Another organisation that came up is the Open Rights Group (ORG, **www.openrightsgroup.org**). This is similar to the US-based Electronic Frontier Foundation, but it operates out of the UK. With UK Prime Minister David Cameron suggesting that the government should have access to all encrypted communication, the ORG has plenty of work to do, convincing politicians that people have a right to private communications in their personal lives.

The ORG runs a number of campaigns to raise awareness of various issues. For instance, "Blocked!" (**www.blocked.org.uk**) scrutinises the UK government's recently implemented web filters, which ostensibly protect children from accessing adult material. But many other websites are being blocked as well, and we should all be concerned that the government has the ability to simply block any site it doesn't like.

Another campaign is Don't Spy On Us (**www.dontspyonus.org.uk**), which notes that the UK intelligence agencies process a whopping 21 terabytes of data every day. ORG is encouraging citizens to protest against unwarranted mass surveillance, and to campaign about the situation.

BALLOT BOX

HELP ORG FIGHT MASS SURVEILLANCE THIS ELECTION YEAR

Join today

and vote for digital rights

TAKE ACTION AGAINST SURVEILLANCE

-SHARE THE CAMPAIGN

The Open Rights Group hopes to make mass surveillance a critical issue during this year's UK elections.

Trisquel GNU/Linux

A distro with no binary blobs in sight.

Trisquel is one of our favourite distributions, even though we don't run it on all our machines. It's an Ubuntu-based distro that takes a no-compromises approach to Free Software: everything is completely free and open. Contrast this with most other desktop-oriented Linux distributions, which include various firmware binary blobs or provide access to non-free software such as Flash and video drivers.

Trisquel takes Ubuntu LTS (Long Term Support) releases and strips out all of the non-free components, producing a very pure distro. It's one of the few distros endorsed by the Free Software Foundation (hence the "GNU/Linux" in the title), and while there are other distros with the same goals, we think Trisquel is the most polished. Its website is slick, the desktop is attractive, and it shares Ubuntu's solid installer. There's no compromise in quality here.

Currently Trisquel is a part-time project for its development team, and is funded entirely by the community. However, the developers accept donations to cover hosting costs, and there's a long-term goal to pay people to work full-time on the project. Having a newbie-friendly completely free distribution is important, we feel, even if most of us use other distros, so Trisquel will be one of the options in the vote.

Trisquel has no commercial backing; it depends solely on donations from the community.

Other projects

Krita, KDE, Inkscape, Code Club...

We had many other great suggestions as well. Some readers asked that we put other graphic and design tools on the list, such as the very awesome *Krita* and *Inkscape*. KDE came up several times – Karl Ove Hufthammer noted that the desktop's developers produce detailed quarterly reports of their activities, showing how donations are used to improve the project (**https://ev.kde.org/reports**).

Or perhaps we could fund the future, so to speak, by encouraging children to take up programming. Code Club (**www.codeclub.org.uk**) provides a network of after-school programming clubs in the UK, targeted at children aged 9–11. This might seem rather young, but then many of us in our 30s and 40s cut our teeth on 8-bit computers such as the ZX Spectrum and Commodore 64. A £120 donation from us would fund a club for 15 children.

Then there's *Ardour*, The Document Foundation, *GCompris*, *GnuPG*, OpenSSL (or one of its spin-offs), science apps, privacy tools and much more. But to finish, we really have to give a mention to HEGX64's idea: "I think some of the money should be spent on porting Systemd to MikeOS". Well, if you insist...

Over to you!

So, those are just some of the potential recipients of our profits. Now it's your turn to vote. We've decided to split up the recipients into two categories: the first is for organisations, and also includes Linux distributions. The second is for software projects. Because our donation pool is £3,000 in total, that's £1,500 for each category. And then we want to have a main winner and two runners up for each category: the winner gets £1,000, second place £300, and third place £200. So in total, six organisations, distros or profits will receive some financial assistance from us. As mentioned earlier, it's not a huge amount of money right now – it's still early days for us as a business, but hopefully as the magazine grows we'll be able to offer more and more at the end of each financial year!

To have your say, go to **www.linuxvoice.com/profitsvote1** and choose your favourites in each category. It's important to us that our readers decide the results, so you'll also need to enter this code to confirm that you've got the magazine: **LV3276XJA**. We will keep the voting open until all readers around the world have the magazine, and then announce the results. Thanks for taking part!

FAQ PULSEAUDIO

Don't run away screaming in terror. Linux audio is easier to understand than the, er, boot system, as these two pages will demonstrate.

GRAHAM MORRISON

Q Hasn't PulseAudio been with us since the dawn of time?

A Yes, it's been around for over a decade. *PulseAudio* has been quietly doing its job for years (we're on the cusp of version 6.0 in February 2015) amid a cacophony of debate surrounding its complexity and effectiveness. And that job is providing to get sound out of your computer.

Q Why have you chosen to cover it now?

A There are huge chunks of Linux that we think need demystifying, especially when the technology is vital and current and still being actively developed. Sound is one of computing's fundamental senses, along with video. We're also certain that lots of users don't appreciate what *PulseAudio* does and why it's so important.

Q Didn't the original Sound Blaster sound card do a decent enough job for sound?

A Ah, the venerable Sound Blaster expansion card – bringing audio to PCs since 1989. Things were simpler then – you'd typically run only one thing at a time, and that one thing would talk to your Sound Blaster directly. If you had a competing Gravis Ultrasound card, for example, you'd have to make sure that whatever you were running supported it. And you also had to worry about IRQ and DMA addresses. Sound is more complex now, but at least we don't have to worry about IRQs.

"PulseAudio allows lots of different applications to share your audio hardware."

Q But why has sound become so complicated?

A We think the best way of explaining this is to use a visual metaphor. Both video and audio suffer from many of the same problems. With video, the solution to running lots of different applications at once is a desktop and window manager. These surround the things you want to run and allow them all to share the same screen. This is what *PulseAudio* does for audio. It allows lots of different applications and processes to share your audio hardware, allowing the user to change their position – or their relative level – in the final output mix. This metaphor can be extended indefinitely: audio sample rate is synonymous with frame rate – 44100 samples per second for compact discs. The number of bits used to store a sample is equivalent to the bit depth of your display. CDs use 16 bits, or 65536 different levels. Most video hardware outputs 24-bit depth – 16.7 million colours. Anti-aliasing, re-sampling, error rates and compression all have analogous causes and solutions in video and audio processing.

Q But what has all this got to do with *PulseAudio*?

A In those years since the Sound Blaster, and we know Linux came a few years later, Linux experimented with all kinds of different audio frameworks – OSS, ALSA, ESD, Jack, GStreamer, Xine and many more. To a greater or lesser extent, they were all trying to simplify audio by hiding the complexity of what they were doing from the user. *PulseAudio* encapsulates some of these frameworks and ideas, and tries to augment them with a modern network transparent framework that's responsive and powerful while remaining simple enough for anyone to use. Mostly, it succeeds and it's only when you need some specific configuration that you notice its complexity, or its inability to play nicely with other audio frameworks. That's when it helps to understand a little of what it's trying to do.

Q What is *PulseAudio* trying to do that's so special?

A If *PulseAudio* had a mission statement, it would be something simple like, "Initiate a sound and hear it." And that's what makes understanding what it does relatively difficult. You hear the results, but you don't see how they've been produced. To be able to perform this one simple task, *PulseAudio* needs to bridge the many layers of Linux audio, some of which we just mentioned, and it starts at the very bottom – talking to the hardware.

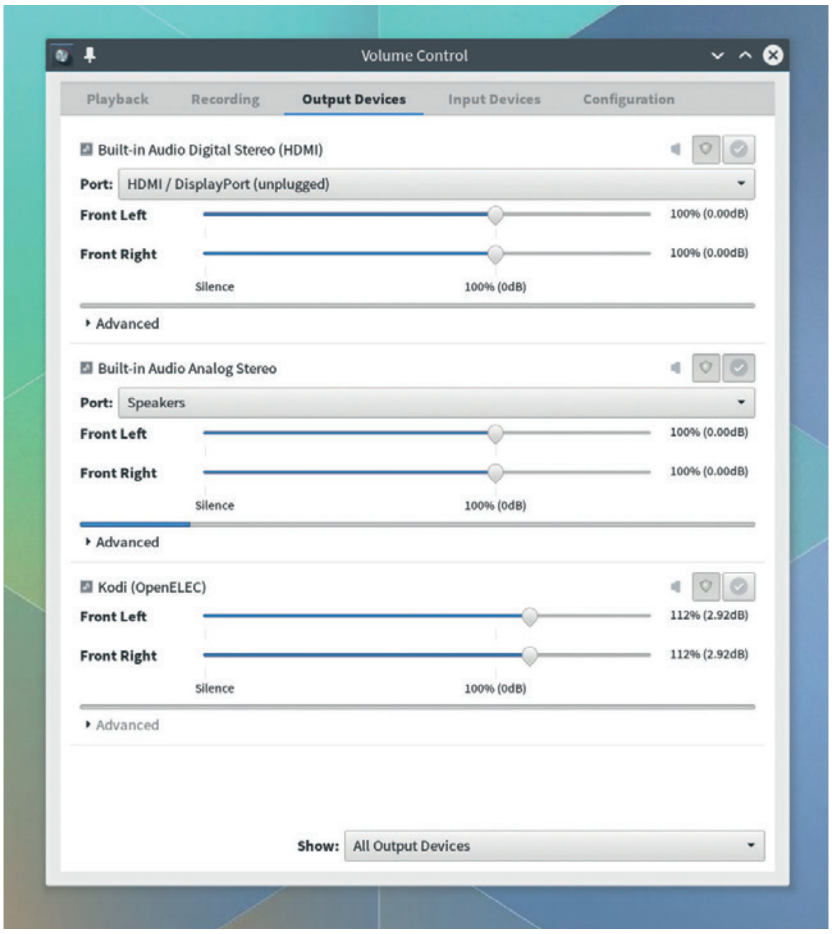
Q Isn't that where you'd normally need a driver?

A Exactly, yes. *PulseAudio* talks to audio drivers directly. But these drivers aren't specific to *PulseAudio*. Instead, it uses the part of the ALSA framework that talks to your audio hardware – the bit embedded within the kernel. ALSA, like *PulseAudio*, bridges several different layers, which may be why there's so much confusion. Above the hardware/driver layer, ALSA is replaced by *PulseAudio*, because only one framework can access the hardware at any one time. That means your audio hardware only needs to support ALSA, as used within the Linux kernel, and it supports *PulseAudio* too. This can lead to confusion as you can't run ALSA alongside *PulseAudio*, at least not without some manual intervention. So while both use the ALSA driver element, only one framework can talk to that driver. If you do want to use ALSA-only, you have to run it through *PulseAudio* or kill of *PulseAudio*'s access to your hardware.

Alongside drivers that talk to your hardware, *PulseAudio* is also capable of talking to your network. This is a little-used feature, but it means that you should be able to use any of your machines running *PulseAudio* as audio input and outputs for another machine on the network. You could have Spotify installed on one machine, for example, and play its output via another machine. It's also the perfect example of how well engineered *PulseAudio* is – the engine is running independently of the input and output hardware.

Q How does this all come together on your desktop?

A On top of the driver layer there's *PulseAudio*'s core. This is where the real work is done, connecting the capabilities of your hardware drivers to the software you're running on the desktop. Using our audio and video metaphor, this is where all the images are combined with the output capabilities and sent on their way. To ensure that *PulseAudio* can be as compatible as possible with older systems, a library layer sits atop the core, allowing applications that know nothing about *PulseAudio* to still play their audio through a compatibility layer. This is how native ALSA applications behave, for example, as the ALSA library is replaced by one that talks to *PulseAudio* instead. But the main desktops are also capable of talking directly to the core and managing which parts of their own sub-system are sending audio. This is what Ubuntu's Unity is doing, for example, and that's why you have a good degree of control over what applications are currently playing back and how loud they are. Gnome is the same.



Built-in Audio Digital Stereo (HDMI)
Port: HDMI / DisplayPort (unplugged)
Front Left 100% (0.00dB)
Front Right 100% (0.00dB)

Built-in Audio Analog Stereo
Port: Speakers
Front Left 100% (0.00dB)
Front Right 100% (0.00dB)

Kodi (OpenELEC)
Front Left 112% (2.92dB)
Front Right 112% (2.92dB)

Show: All Output Devices

Get much better control over *PulseAudio* with the *pavucontrol* mixer.

Q This is great, but it still doesn't give me access to all that power you've been talking about.

A The ultimate in *PulseAudio* control comes from the command line, and specifically, a command called **pactl**. For example, typing **pactl load-module module-raop-discover** will load a dynamic network discovery module that will detect any AirPlay devices on your network (as long as **roap** is compiled into your version of *PulseAudio*, which it should be). *Pavucontrol* should now list any AirPlay devices, such as Kano or *XBMC*, and allow you to select these as outputs, enabling you to play music from one machine that outputs on another, and there are many more different modules and combinations.

Q This sounds straightforward, should I be worried if *PulseAudio* no longer scares me?

A Not at all. When it works it's brilliant and you don't have to worry about it. But hopefully, you've got some appreciation of its complexity and how it's performing all this magic, as well as knowing it can be used for some advanced trickery too. If you want to know more about the advanced stuff and grab version 6 as soon as it's been released, *PulseAudio* lives at **www.freedesktop.org/wiki/Software/PulseAudio/** LV

JONO BACON

He's the Pharaoh of community management, but what led him to switch from Ubuntu to the millionaire philanthropic prize fund, XPRIZE?

Jono Bacon is the co-founder of LUG Radio, LUG Radio Live and the brilliant Bad Voltage (**www.badvoltage.org**) podcast. He was also the Ubuntu Community Manager at Canonical for eight years, providing a bridge between the company and its community of users, while at the same time writing books about community and conflict, as well as founding the Community Leadership Summit. Last summer, he switched streams to join the XPRIZE Foundation as its Senior Director of Community. We sent Graham Morrison to find out where the crossover might be, and what open source and XPRIZE could learn from each other.

LV Has it felt like a natural progression, going from Canonical to XPRIZE?

Jono Bacon: I think so. When I joined XPRIZE and started to think about building a new community that is designed to create a brighter future, the first thing I thought was that communities share many of the same principles, so I could pull from much of my existing experience.

The XPRIZE ethos is based on solving problems with technology, and that technology grows at an exponential rate. I then realised that communities have this exponential growth curve in many ways too – we saw that with Wikipedia, we saw that with open source, we've seen it politically with the Arab Spring. I never really realised that communities were an exponential entity. With that in mind, when we start socialising that as a concept, we can define these broader goals. Much of this is inspired by open source and my work at Canonical, but I think we can actually do way more in the world in new and different areas.

LV Is this is the emergent philanthropist in Jono Bacon?

JB: [laughter] Worst philanthropist ever! I believe that technology can solve all these major problems but I also believe that communities are a critical part of solving these major problems too. They just need to be better organised. We've been through the dark ages of community management, now we're starting to see people write it down – we're in the Renaissance period. My job at XPRIZE is obviously focused on the XPRIZE community, and growing that out. But in a broader sense as well, I want to get people to think about what we can do with communities and how well structured and organised communities can be world-changing.

LV Will exponential growth happen outside of social media? That's a huge challenge.

JB: When I was thinking of going to XPRIZE, I watched this talk that Peter [Peter Diamandis – Chairman and CEO of XPRIZE] did in the Arab Emirates. He said that as human beings we think in this local and linear way, because our brains are designed that way, because a thousand years ago the changes in technology or the surroundings between one generation and another were very very small. But even the difference between my dad's generation and my generation is huge already, if we think about what's happened.

Take the maker movement, for example. Chris Anderson, who used to edit *Wired*, has written this book where he talks about his granddad moving from Switzerland, and he basically invented an automated sprinkler system in his garage. He spent some money on getting it patented, and he sold it to a company and it made a little bit of money. But it didn't really have that much of an effect, and Chris wondered whether he could do the same thing using 3D printing, Arduino, and other pieces.

So he tried to do what his grandfather did, but designed for today. He did it for a few hundred dollars. His sprinkler system would listen to **weather.com** and based upon that would apply the relevant level of moisture. A lot of the pieces were already there for him because a community was already trying to automate ways of growing weed [laughs] more effectively. So there's this massive community of people who are building these advanced sprinkler systems basically driven by hydroponics. What was interesting was that he could build something unique because a lot of the pieces were created by different communities of people coming together. And that's the reason why the community piece is so important.

LV Will you have room in your new job to do that?

JB: With an open source company, what a community looks like is fairly well defined. You have the technology and you get people involved in building that technology. What's different with XPRIZE is that there are going to be lots of smaller communities because we have lots of different areas of focus, from oceans to space to education to life sciences and more.

The work I am doing is to provide an on-ramp for anyone to have a practical effect in changing the world. This includes people writing code, running local XPRIZE Think Tank groups, designing user interfaces for teaching kids literacy and more. What is fun about this is that it's a totally new community. The job was pitched to me by Peter and he said, "I want you to transform our progress at XPRIZE ten-fold with communities." We didn't get into a lot of the details about how exactly how it would work: he is the visionary, the big thinker, and he wanted

"Technology can solve all these major problems, but communities are a critical part of solving them too."

me to come and disrupt our normal way of working at XPRIZE to explore how we make it more collaborative and so anyone can play a role in building this brighter future.

LV That's probably the best way...
JB: It's great but it's a bit scary because this is all new and success is not assured. We just don't know exactly what it's going to look like. Right now we are starting with some logical building blocks and I want to connect my prior experience to see what makes sense and what interesting ideas the community comes up with for areas where people can collaborate and get involved. Importantly, our community will play a key role in what path we choose to follow; I want this to be a really collaborative experience.

LV It could be a great opportunity.
JB: Exactly. But it's challenging. I found out from one of my new colleagues, who is on the Google Lunar XPRIZE to get a drone on the moon...

"There's a mantra at XPRIZE: 'The day before something is a breakthrough, it is a crazy idea".

LV That's just so cool, the fact that you can just drop that into the conversation…

JB: I know [laughs]… well, I said to my colleague that one of the things I want to do is set up a forum in which teams can collaborate together on areas of common interest, instead of all of them solving the same problems over and over again. It would be interesting, for example, to bring industry experts into the discussion to better support the teams, and he said, that's doable with certain types of space research, but not all. The challenge today with space is that if you're building space technology over a particular size, then the government makes you restrict how that information is shared; they consider over a certain size to be creating weaponised space technology.

It's very difficult to build collaboration within that world, when the government locks it down. But then that doesn't apply to other areas such as oceans, literacy or whatever else. I went into it initially thinking we needed to create one massive XPRIZE community, like we did with Ubuntu. Now I'm thinking a little differently about that. I think it will manifest in lots of different areas.

LV I guess you could start with small ideas and see which gain any momentum?

JB: Exactly. There's so much we can do. I know so many smart people that aren't currently connected. And when we connect them together, it is going to be pretty amazing.

LV What do you think the open source communities could learn from XPRIZE communities?

JB: The thing that XPRIZE does that I think is really interesting is evolve how you define incentives for solving these issues. The prize development process is really expansive. There's a lot that goes into it. It is not just making sure that the technology exists in the first place; you want to look at the technology where we are today, predictions around where technologies are going in the next 3–5 years…

LV Is there a laboratory with charts somewhere?

JB: Yeah! They bring in industry experts, consultants, scientists, business leaders and more. They have these multiple phases about how a prize goes from being an idea to something more.

There are ideas around weather control, building flying vehicles, curing diseases, producing better cleanup technologies, and more. Each idea goes through this process, but it's not just about if technology will get you there – is there an interest there? Does it affect hundreds of millions of people? The goal here is not to build cool gadgets; it is to bring profound impact to the world.

But it is also about creating new industries. Creating great R&D is nothing if we can't get it to the market and get people competing to drive prices down. This is a big chunk of the picture too.

A lot of that knowledge of the challenges in the world, and what we can do to fix them, can bubble out in so many different directions. Some of the work that's gone into XPRIZE may provide inspiration for where developers may want to focus their efforts, for example, on software that could be used as part of these solutions.

LV It sounds like a dream job.

JB: It is pretty neat. When I started my career, I wanted to focus my efforts on having a contribution to something that has a wider-reaching benefit to the world. Canonical was a great place to do this focusing on building free and open technology that empowers people to educate themselves, start small businesses, create art, collaborate, and more.

I wasn't really looking to leave Canonical, but when this thing came up – I'd heard of XPRIZE, but I didn't really know what it was, so I started looking at it and thought 'wow.' There's a lot that could happen here, a lot of potential, but a lot of new and culturally different work to do. It seemed like the right mix of a great opportunity but also a real challenge to get my teeth into.

What's the most important lesson you've taken from Canonical?
JB: I think there's probably two things; one thing that Canonical really helped me with is to think in a very strategic way. To really think about what we want

"The goal here is not to build cool gadgets; it is to bring profound impact to the world."

to do. Where do we want to go? What's it going to look like? I learnt a lot of project and people management from some amazing people.

The most important thing for a lot of this stuff is having the passion, and one of the things I learnt most from Mark [Shuttleworth – Ubuntu founder and Earth's second space tourist], is not that we're going to "try" to do this, we're going to go out and "do it". Having that level of confidence but not arrogance, and not dominance, is critical. Ubuntu doesn't go out to crush the competition, it's out to be better than the competition, and I think that's the right kind of approach, and I learnt a lot from Mark in that regard. He's got an unbending level of positivity and forward momentum – one of the hardest things about leaving Canonical is not working for Mark any more.

Are you still using Ubuntu?
JB: Yeah, but I do have to use a Mac for work, because XPRIZE has a small IT team and a limited set of platforms they can support within the remit of that team. I use Ubuntu for everything else and I am working to get Ubuntu into XPRIZE more.

How do you think your ebook (*Dealing With Disrespect*) has been received? It was a brave thing to do; stating your case can often be asking for trouble and it's so much easier to just keep your head down.
JB: The thing I was most scared about in writing that book was that people might interpret it as tolerating bullying. I don't think it's about that. The book touches on a big source of conflict which I think are the different ingredients that make up a human being. Age, culture, gender – all those different pieces. As we all know, when you start talking about those kinds of areas, it doesn't matter what you say, some people are going to read it in the way they want to.

Was there anything in particular that pushed you to the point of writing it?
JB: Yeah. I can tell you exactly what it was, and this is going to sound a bit weird actually. I bought a Playstation 4 – I was sadly one of those people who lined up outside of a Best Buy at 7 o'clock in the morning – what a nerd. There's a feature on Playstation 4 called Playstation Live, and people can stream videos live. I'm a big Reddit fan and I was following the Playstation 4 subreddit and there was post on there at the top that said there was a guy live streaming right now who's drunk and he's just kicked his dog and he's hitting his wife or girlfriend – and the post said "He's live streaming now, go and check it out."

So I went to check it out and he was surrounded by cans of Bud Lite and he was on the phone – this is a little bit funny – but basically, he gave his phone number out on the live stream so people were prank calling him pretending to be this MMA fighter or something, Reddit had conspired around this. On the one hand it was funny, and people were egging him on.

What was clear to me though was that this guy was having a breakdown. I started typing in, "Turn the Playstation 4 off. Go to bed and sober up. Just, stop this." And people started giving me grief, saying, "What the hell is wrong with you. This is brilliant, blah, blah..." People were insulting me and I was just trying to be a good citizen.

I was really angered by this. These people are absolute idiots, and it got me thinking about the conduct of people on the internet and how mean spirited people can be to serve their own interests and opinions. And that was the night I had the idea for the book. I had the benefit of being the community manager for a large open source project and I'd learned a set of things that are now in the book. Most people don't go through that experience so they've not had that opportunity to learn how to put things in context and develop a thicker skin.

There's some weird voyeurism going on. They've got a games console and they're sitting with a beer watching other people watching other people.
JB: One of my goals for the book is that when someone's in that position and someone has said something mean, and it dings that person's confidence, I want someone else to say, "You should read this book. It's really quick and easy – you can read it in a couple of hours. That will help." And that's the reason why it's free to download and it's available online. I hope it helps people stay focused and positive on creativity and not negativity.

Jono says the worst thing about moving from Wolverhampton to California is being so far away from the Black Country Living Museum.

INSIDE THE fsfe

Mike Saunders and **Graham Morrison** popped by the FSFE head office in Berlin to see how the organisation is spreading the word about FOSS.

You've almost certainly heard of the Free Software Foundation before. This is a US-based non-profit organisation set up by Richard Stallman, the creator of GNU, in 1985. Originally it was established to fund programmers, but over the years it has moved into other realms, handling legal issues and promoting Free Software.

Since 2001, a European spin-off has existed: the Free Software Foundation Europe (FSFE – **www.fsfe.org**). It's a sister organisation of the FSF, but also a financially and legally separate entity. On a recent trip to Berlin, we poked our heads into the FSFE office to see how they work on a daily basis.

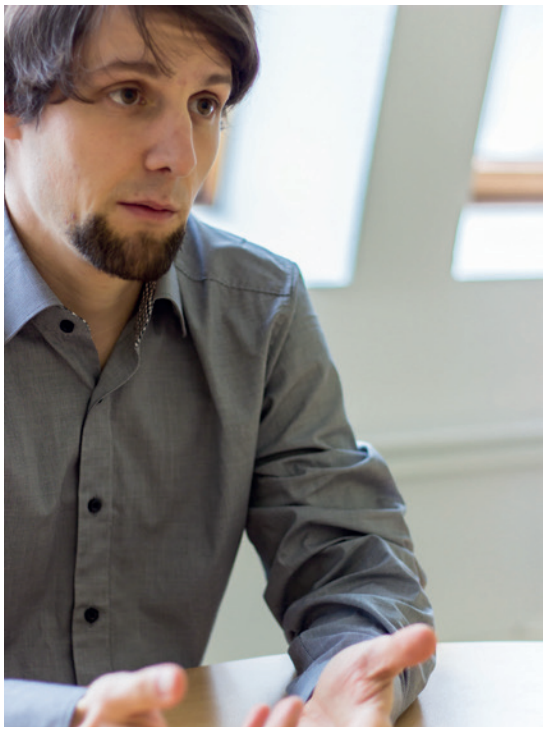
"We have to fight for our freedoms... but it's important to recognise your successes as well."

We spoke to Matthias Kirschner, the Vice President, and asked him what the organisation does: "Along with promotional materials like leaflets, which we provide to people who want to spread the word about Free Software, our other activities include lobbying work. We go to Brussels, participate in meetings, talk to people in the parliament and the European Council. We explain digital sovereignty, that you should control your own infrastructure and talk about device ownership."

Dealing with politicians is tough, though, especially when they're not technically inclined. Matthias described how in the early 2000s, when he approached politicians to talk about Free Software and related issues, he felt he was often dismissed. "They would think: who are these freaks, talking about source code and digital rights and stuff?" But as time went on, as more and more people started approaching their local politicians and MEPs about these issues, they had to take it more seriously – or at least, ask their staff to find out what it's all about.

Along with lobbying, the FSFE works to distribute materials that help people to understand Free Software and the importance of having control of your own data and devices. The Snowden revelations have been particularly important in this respect, and in recent years the FSFE has moved from preaching to the choir – that is, telling people who already use GNU/Linux about the importance of Free Software – to spreading the word among the uninitiated:

"We have started to go to more events where Free Software is usually not represented. For instance, our Munich group has been to two street festivals, where they have a pavilion and give out information about Free Software. We hand out thousands of leaflets at street festivals in Düsseldorf too. Our Vienna group goes to a big game conference – they wanted to have more general information about Free Software, for people who don't know what it is at all."



When Citizenfour (a film covering the NSA spying scandal) hit the cinemas in Berlin, FSFE supporters stood in front of the cinemas, handing out leaflets explaining the importance of email encryption and how to use *GnuPG*. In one cinema in the UK, these leaflets were also handed out when people bought tickets, so this targeting is becoming increasingly important to the FSFE.

This good work can only happen if the organisation is well funded. Thankfully, it is. "Around one third of our donations come from our Fellowship", says Matthias, "those are sustaining members. The rest is from company donations or one-time donations from individuals. Sometimes we also have EU projects that provide funds. Then there are speaker fees and merchandise, but the main thing is donations."

Join us now and share the software

We were also curious to learn about Richard Stallman's involvement in the FSFE today. Matthias explained that for daily jobs and short-term projects, they don't coordinate with him all the time. "But sometimes Richard sees something and gets involved. Like, we originally had a Free Your Android page on our website. It was just a simple wiki page. But Richard emailed and said: 'This is a nice campaign, but you should change X and Y and Z, and that would make it better'."

But what would happen if Stallman were to step down? Matthias doesn't think this is likely any time soon – he may go slower as the years roll on, but his passion and enthusiasm are as strong as ever. But Matthias also noted that Free Software has become much bigger than just a single personality, and the principles are what really drives the movement now.

At the end of our visit, we asked Matthias if he was happy with what the Free Software Foundations have achieved so far. He noted that a lot has been done, but it's a long-term battle, comparing it to freedom of the press. That didn't come in overnight, but was a long process and has to be maintained and fought for. Free Software is doing well, but to get it into more government, schools and other public institutions, a long slog is required.

"We have to fight for our freedoms, again and again and again. But it's important to recognise your successes as well. Look at all the devices around the world running Free Software. Look at all those companies that thought Free Software was a virus in the IT field, and now they publish their work under Free Software licences. Lots of companies now talk about Free Software as if they invented it!" LV

FSFE's Berlin Office
For the 2014 European elections, the FSFE created the "Free Software Pact" a document that candidates for the European Parliament could sign saying that they favour Free Software and will protect it from potentially harmful legislation (**www.freesoftwarepact.eu**). 162 candidates signed up, and 33 were elected.

UNPROTECTED COMMUNICATION

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

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

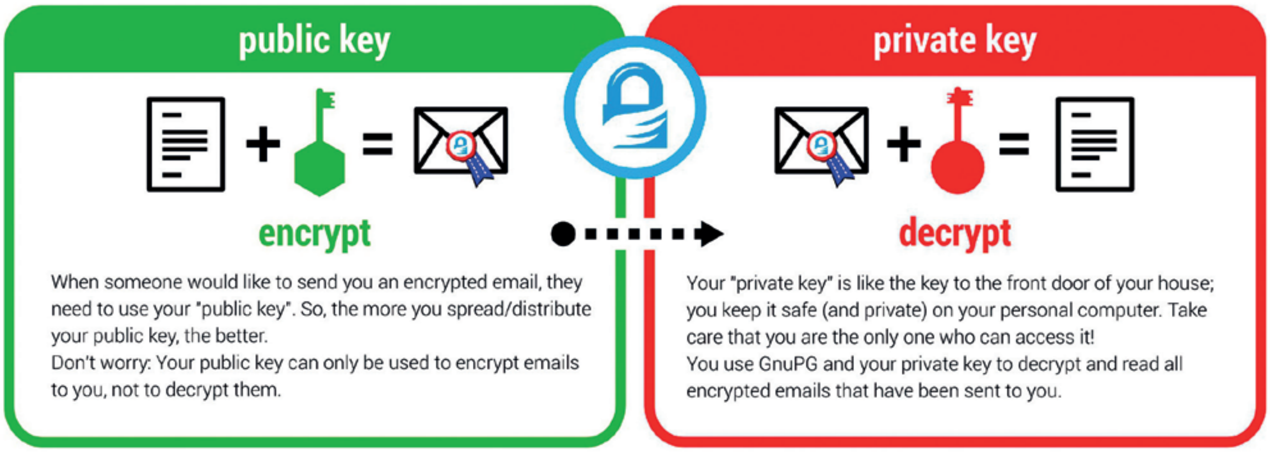
GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

About GnuPG

Thousands of people already use GnuPG, for professional and private use. Come and join us! Each person makes our community stronger and proves that we are ready to fight back.
Whenever an email that is encrypted with GnuPG is intercepted or ends up in the wrong hands, it is useless: Without the appropriate private key it cannot be read by anyone. But, for the intended recipient - and only for her - it opens like a totally normal email.
Sender and recipient are both safer now. Even if some of your emails contain no private information, consistent use of encryption protects us all from unjustified mass surveillance.

What makes GnuPG secure?

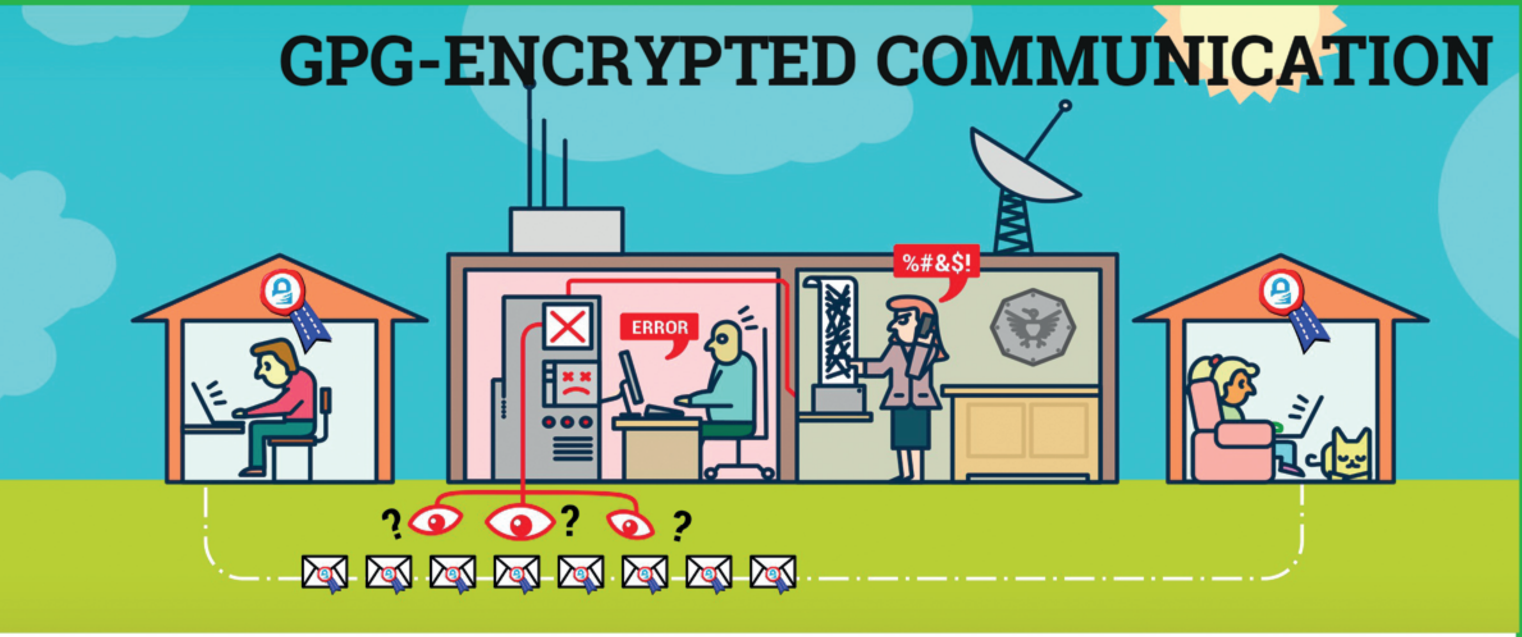
GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.
If no one is allowed to see the source code of a program, nobody can be sure that it does not contain undesirable spy programs - so-called "backdoors". If software does not reveal how it works, we can merely trust it blindly.



When someone would like to send you an encrypted email, they need to use your "public key". So, the more you spread/distribute your public key, the better.
Don't worry: Your public key can only be used to encrypt emails to you, not to decrypt them.

Your "private key" is like the key to the front door of your house; you keep it safe (and private) on your personal computer. Take care that you are the only one who can access it!
You use GnuPG and your private key to decrypt and read all encrypted emails that have been sent to you.

GPG-ENCRYPTED COMMUNICATION



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

Practical advice

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

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

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

About the fsfe

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

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

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

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

What is Free Software?

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

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

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

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

fellowship of fsfe

YOUR AD
HERE

LV

Email andrew@linuxvoice.com to advertise here

LINUXVOICE REVIEWS

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

Andrew Gregory
Wise man say, the best way to start a new project is to finish the old one.

The selection of reviews over the next few pages illustrates a truism of Linux and Free Software: choice is good. Like the verse on Ozymandias' plinth, these words will reach down through the generations long after all else is dust. Choice gave us a fork of OpenOffice. Choice gave us OpenOffice in the first place, when the pragmatic majority would just have stuck with Microsoft Office.

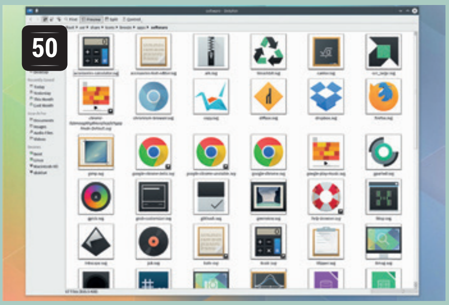
Choice gives us the CubieBoard. Surely there's no point launching an ARM board in the teeth of the 4.5 million-selling Raspberry Pi? Oh but there is – with a slightly different focus on what users want, this little device will carve out its own nice, and the two will live happily side by side.

Let a hundred flowers bloom

Icaros may seem pointless to many. But its ultra-low system requirements mean that, despite its roots in a 20-years defunct OS, it could yet get a new lease of life in embedded, mobile or otherwise Internet of Things appliances. And KDE? Once dominant, it lost ground by failing to listen to its users; it had to fight back to survive, and has now improved beyond measure to reclaim its position as king of the desktops. That's what choice is about. We're lucky to have it.

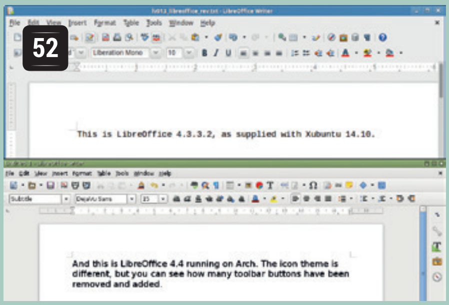
andrew@linuxvoice.com

On test this issue...



KDE Plasma 5.2

Ardent Prince fan **Graham Morrison** plays with the desktop environment formerly known as plain old KDE.



This is LibreOffice 4.3.3.2, as supplied with Xubuntu 14.10.

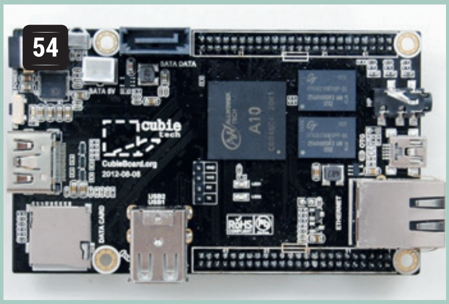
And this is LibreOffice 4.4 running on Arch. The icon theme is different, but you can see how many toolbar buttons have been removed and added.

LibreOffice 4.4

The Free Software world's favourite office application is back, and **Mike Saunders** is here to put it through its paces.

Icaros Desktop 2.0.3

Mike Saunders investigates a retrostalgic alternative desktop experience born out of the Amiga OS of the early 90s.



CubieBoard A20

ARM-based boards are all the rage at the moment. **John Lane** tests one with huge amounts of potential.

BOOKS AND GROUP TEST

Lawrence Lessig is a man well worth listening to, and his book *Free Culture* is a great read. On the one hand it's about copyright law; on the other, it's a story of the way in which big companies are monopolising (ologopolising?) parts of our culture that have traditionally always belonged to everyone.

In the Group Test we're looking at the BSDs. This Unix-derived operating system very nearly grew to become the operating system that Linux is today, but a historical quirk has largely kept it in the server room. Spin up a virtual machine and try one out!

CLIPPING THROUGH
ONE MAD WEEK IN VIDEO GAMES
LEIGH ALEXANDER
FREE CULTURE
HOW BIG MEDIA USES TECHNOLOGY AND THE LAW TO LOCK DOWN CULTURE AND CONTROL CREATIVITY
LAWRENCE LESSIG
Raspberry Pi FOR DUMMIES
Sean McManus
Mike Cook

KDE Plasma 5.2

Graham Morrison finds another way of spreading the good word about KDE – reviewing a release he can recommend.

DATA

Web
kde.org
Developer
Team KDE
Licence
LGPL/GPL

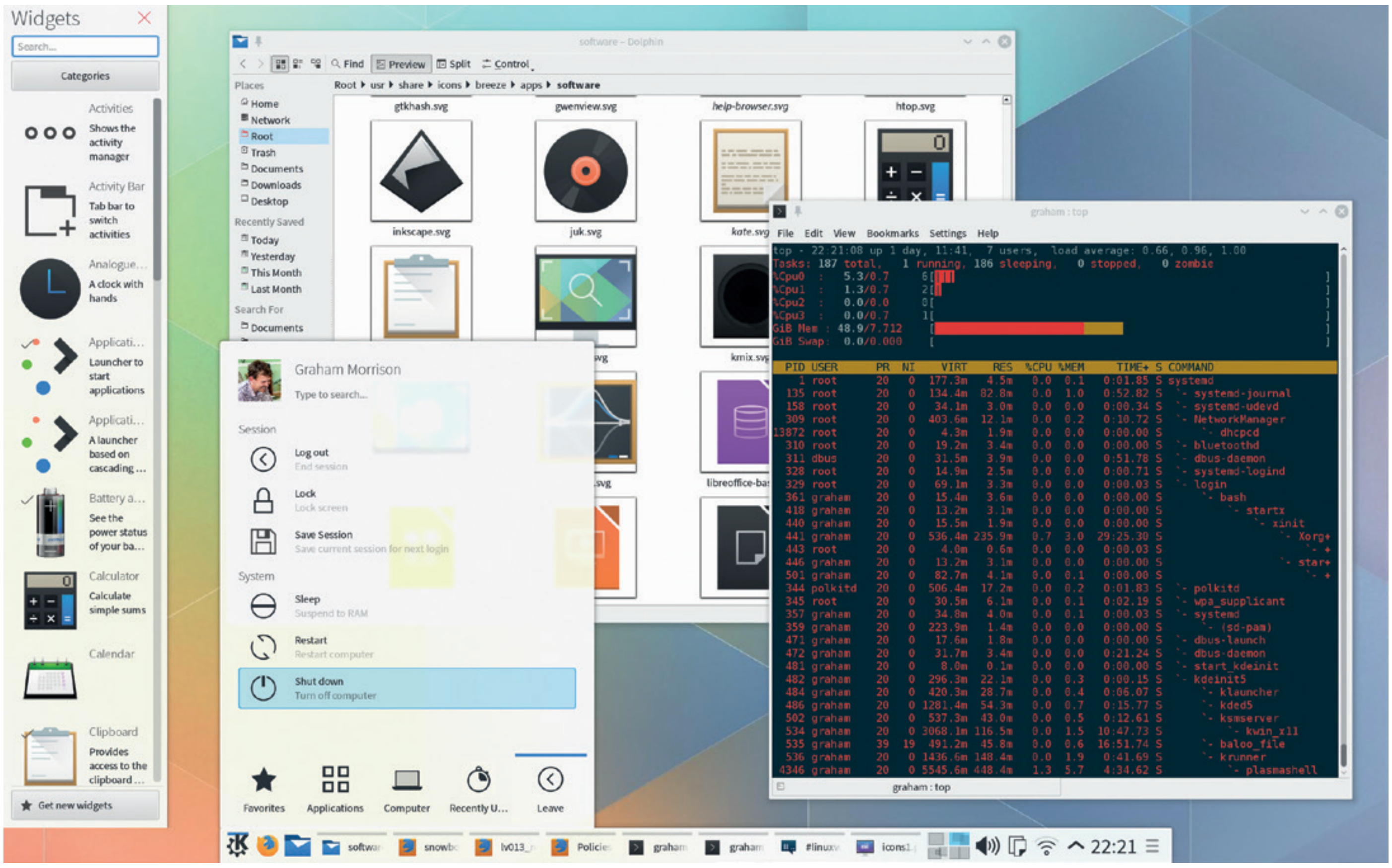
The KDE desktop, or KDE Plasma 5 as it is now known, is developing at quite a pace. The major 5.0 upgrade came in August 2014 with this major update following less than six months later. In the past, this could signal that stability has been thrown out with the demands of adding new features. But not this time. We're very happy to report that the one new feature in this new release that you can't see – it's stability – is the best reason for making the upgrade. After weeks of running KDE Plasma 5.2 on our two main machines, we've experienced fewer crashes and stability issues than with any previous KDE release. The panel does occasionally crash, and a few wayward widgets can force you to restart Plasma manually, but nothing caused us any loss of data or complete loss of our desktop session.

Other than the sisyphean chore of bug squashing, a huge amount of effort has been expended polishing the visuals. Not only does this make everything far prettier, we think it's a great sign that the APIs have reached maturity and that the developers have enough confidence in their capabilities to start fleshing out Plasma's appearance. If you've got a high-resolution display, the first you'll notice of this are the system tray icons in the panel. These show your Wi-Fi signal or highlight any notifications. So too are the global icons you find in the launch menu. They're now all monochrome, pristine and perfectly scaled, as befits their vector sources.

To go along with these, there's an entirely new icon set which has been designed to complement the minimal aesthetic. Design is always going to be subjective, but we love the new designs. They're abstract and colourful with a kind of solarized 1980s neon palette that gives an overall impression of lightness and simplicity. Even better, there are dark versions for those of us who prefer a reversed colour palette. In-keeping with appearances, we also really like the Breeze window decoration that retains the air of minimalism while injecting a little character into your desktop.

Warm and GUI

These icons also scale perfectly in the launch panel, which is brilliant for higher resolution or high-DPI displays, although the same doesn't happen for the bitmaps used by the task manager applet. We're also slightly bemused by the lack of a higher resolution version of the desktop background image. The panel itself has had some refinement, with the main feature



```
top - 22:21:08 up 1 day, 11:41,  7 users,  load average: 0.66, 0.96, 1.00
Tasks: 187 total,   1 running, 186 sleeping,   0 stopped,   0 zombie
%Cpu0  :    5.3/0.7      6[                                                    ]
%Cpu1  :    1.3/0.7      2[                                                    ]
%Cpu2  :    0.0/0.0      0[                                                    ]
%Cpu3  :    0.0/0.7      1[                                                    ]
GiB Mem :  48.9/7.712     [                                                    ]
GiB Swap:   0.0/0.000     [                                                    ]

  PID USER      PR  NI    VIRT    RES  %CPU %MEM     TIME+ S COMMAND
    1 root      20   0  177.3m   4.5m   0.0  0.1   0:01.85 S systemd
  135 root      20   0  134.4m  82.8m   0.0  1.0   0:52.82 S  `- systemd-journal
  158 root      20   0   34.1m   3.0m   0.0  0.0   0:00.34 S  `- systemd-udevd
  309 root      20   0  403.6m  12.1m   0.0  0.2   0:10.72 S  `- NetworkManager
13872 root      20   0    4.3m   1.9m   0.0  0.0   0:00.00 S      `- dhcpcd
  310 root      20   0   19.2m   3.4m   0.0  0.0   0:00.00 S  `- bluetoothd
  311 dbus      20   0   31.5m   3.9m   0.0  0.0   0:51.78 S  `- dbus-daemon
  328 root      20   0   14.9m   2.5m   0.0  0.0   0:00.71 S  `- systemd-logind
  329 root      20   0   69.1m   3.3m   0.0  0.0   0:00.03 S  `- login
  361 graham    20   0   15.4m   3.6m   0.0  0.0   0:00.00 S      `- bash
  418 graham    20   0   13.2m   3.1m   0.0  0.0   0:00.00 S          `- startx
  440 graham    20   0   15.5m   1.9m   0.0  0.0   0:00.00 S              `- xinit
  441 graham    20   0  536.4m 235.9m   0.7  3.0  29:25.30 S                  `- Xorg+
  443 root      20   0    4.0m   0.6m   0.0  0.0   0:00.03 S                      `- +
  446 graham    20   0   13.2m   3.1m   0.0  0.0   0:00.00 S                  `- star+
  501 graham    20   0   82.7m   4.1m   0.0  0.1   0:00.00 S                      `- +
  344 polkitd   20   0  506.4m  17.2m   0.0  0.2   0:01.83 S  `- polkitd
  345 root      20   0   30.5m   6.1m   0.0  0.1   0:02.19 S  `- wpa_supplicant
  357 graham    20   0   34.8m   4.0m   0.0  0.1   0:00.03 S  `- systemd
  359 graham    20   0  223.9m   1.4m   0.0  0.0   0:00.00 S      `- (sd-pam)
  471 graham    20   0   17.6m   1.8m   0.0  0.0   0:00.00 S  `- dbus-launch
  472 graham    20   0   31.7m   3.4m   0.0  0.0   0:21.24 S  `- dbus-daemon
  481 graham    20   0    8.0m   0.1m   0.0  0.0   0:00.00 S  `- start_kdeinit
  482 graham    20   0  296.3m  22.1m   0.0  0.3   0:00.15 S  `- kdeinit5
  484 graham    20   0  420.3m  28.7m   0.0  0.4   0:06.07 S      `- klauncher
  486 graham    20   0 1281.4m  54.3m   0.0  0.7   0:15.77 S      `- kded5
  502 graham    20   0  537.3m  43.0m   0.0  0.5   0:12.61 S      `- ksmserver
  534 graham    20   0 3068.1m 116.5m   0.0  1.5  10:47.73 S          `- kwin_x11
  535 graham    39  19  491.2m  45.8m   0.0  0.6  16:51.74 S      `- baloo_file
  536 graham    20   0 1436.6m 148.4m   0.0  1.9   0:41.69 S      `- krunner
 4346 graham    20   0 5545.6m 448.4m   1.3  5.7   4:34.62 S          `- plasmashell
```

We experience a couple of glitches – such as option selection in the launch menu – but this is the most stable release of KDE yet.

Inside Digia and Qt 5.4

KDE Plasma 5.2 is built on Qt, can be run on either version 5.3 of Qt, or version 5.4, which was released in December 2014. We had a chance to visit the Qt/Digia offices in Berlin at the same time, where were able to talk with Maurice Kalinowski, one of Qt's first two developers in Germany back in 2006.
We asked how he balanced the commercial interests of Digia and Qt with the obligation that Qt has to open source and the KDE project.

"Everyone here in the company is an open source advocate, but clearly we also need to see how we can make a living out of that. The general idea is that, at least with the basic offering, everyone should be able to use what we do," Maurice told us, adding that this means the project is committed to both the GPL and the LGPL licences currently used. Any licensing changes come afterwards, "when it comes to very tailored user cases or very deep core user cases – when it comes to deep profiling tools, for example. We then make an offering that is then attractive to somebody." Maurice is referring to the QML compiler, first released in Qt 5.3 only for paying users of Qt, along with an additional profiler that's being developed to speed up those users' code.

The non-free nature of these tools has caused some criticism, with Aaron Siego saying in May 2014 that he hoped there would be an open source version by the release of 5.4, or that the community may have to provide one. But other Digia/Qt initiatives, such as the open governance model, have had a positive effect on Qt and its relationship with KDE. Maurice told us that this in particular has helped KDE get closed to Qt core, simplifying KDE's own libraries and building a better experience for everyone.

We had a quick chat with the Senior Manager of *Qt* tools, Maurice Kalinowski, just as *Qt 5.4* was being unleashed onto the world.

being the ability to undo the removal of a widget. We appreciate this feature, but we feel it highlights a problem with panel and widget placement in general. It's difficult to hit the target when you're click-dragging something, and the addition of an undo only mitigates this issue rather than providing a genuine solution. The new 'Add Widgets' window, for example, is now shown vertically against the left edge of your screen, and it's a long click and drag to install a widget from the top-left of your display to the bottom-right of your panel.

Widget creation is now officially based on QML 2, and older 4.x widgets will need porting. This has left Plasma with very few widgets to choose from, and the skeletal list that greets you when you click on 'Download New Plasma Widgets' is very disappointing. The technical documentation that should describe how to create your own widgets is also lacking, with only the guide to their abstract structure currently visible at **techbase.kde.org**. We wanted to create our own 'show the date' widget, which should be the simplest of the simple things to create. But with a missing tutorial and no real guidance, we had to postpone our plans. The lack of decent guidance is also going to hinder adoption.

With each new release, more of KDE's native applications get ported to the new frameworks. Old applications aren't a huge issue because it's now perfectly feasible to run both KDE 4 and KDE 5 applications side by side with no noticeable side-effects, but it does mean newer applications can take advantage of all the new developments. These include *Kate*, *Konsole*, *Muon* and *Gwenview* alongside new tools for managing Bluetooth devices and SSH keys in this release. There are many more that need to be ported, but this is firm progress and when you check the *Git* repositories for many of the other components, you'll often find versions using the newer frameworks. The old *KDM* login manager has been replaced by the long-recommended *SDDM*, which now includes its own configuration panel. And the reordering of results in our default application launcher, *Kickoff*, has finally made it usable again.

This is the first KDE Plasma 5 release we'd recommend to avid KDE 4.x users, because the appearance, stability and refinements are worth braving the missing elements. More conservative users will be better off waiting for a proper audio mixer applet and a fully populated configuration panel and file manager, but if you enjoy using KDE for the sake of it, this is the strongest release yet and one that fills us with excitement for the future of its development.

LINUX VOICE VERDICT

A great release that gets us excited about the next 18 months of KDE development.

★★★★☆

LibreOffice 4.4

It's the "most beautiful" release yet, according to the developers, but is it more usable? **Mike Saunders** investigates.

DATA

Web
www.libreoffice.org
Developer
The Document Foundation
Price
Free (open source licences)

Over the years, we've had a love-hate relationship with *LibreOffice* (and its previous incarnation as *OpenOffice.org*). We love it because it's a hugely versatile and well-integrated office suite that does 99% of typical jobs that people use *MS Office* for, and is therefore a great way to wean people off Microsoft's wares. We use it extensively in making this magazine – not just in writing articles, but in managing our subscriber database and other non-editorial jobs.

So any improvement is most welcome, and *LibreOffice 4.4* – described as "the most beautiful version ever" by The Document Foundation – is one of the most ambitious releases we've seen in a while, with some fairly significant changes to the interface. No, *LibreOffice* hasn't adopted the "ribbon" menu of Microsoft's suite, but the toolbars have been redesigned to remove rarely-used buttons and add quick access to other features.

In *Writer*'s top toolbar, for instance, the "document as email" and "edit file" buttons are gone, while buttons for adding page breaks and comments have been added – very sensible decisions, we feel. The second (formatting) toolbar has also been redesigned, with easy access to extra formatting such as strikethrough, superscript and subscript. Styles in the drop-down list can now be edited in place, while a sidebar is enabled by default to provide quick access to properties for the current character or paragraph, along with a document navigator and image gallery.

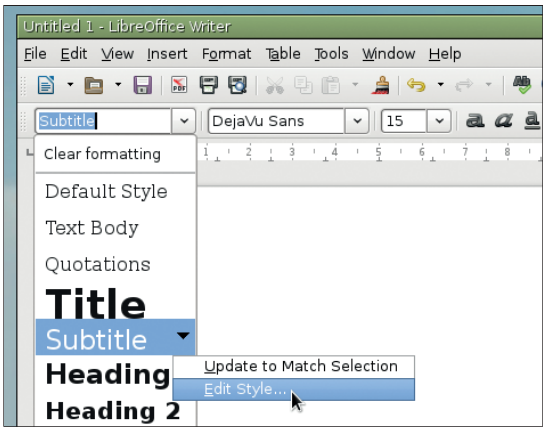
Ch-ch-ch-ch-changes

Calc and *Impress*, *LibreOffice*'s spreadsheet and presentation components, have seen overhauls of their toolbars and interfaces. And in general, we're happy with the changes: they're significant enough to make the suite more pleasant to use, especially for newcomers, but they're not too invasive that long-time users will feel like it's an entirely different program.

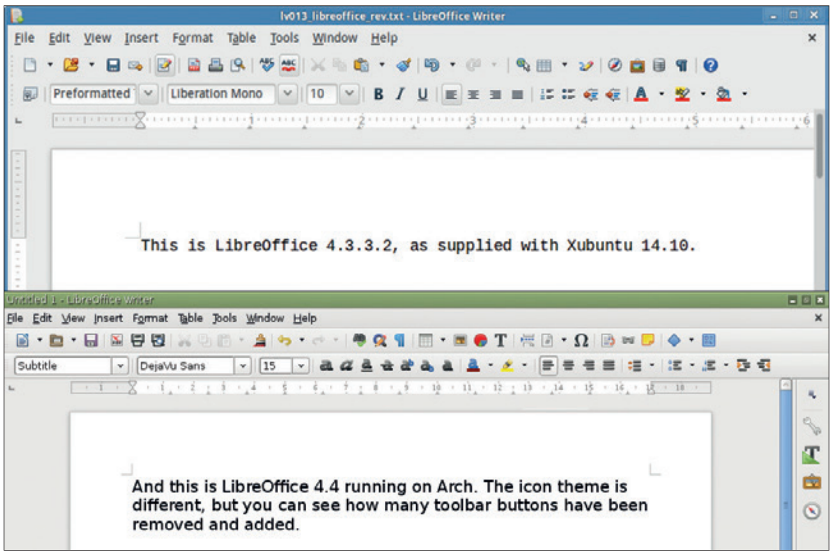
But there's a lot more than just cosmetic changes. Document import and export has improved greatly: there are new import filters for *Adobe Pagemaker*, *MacDraw* and *RagTime* (a German DTP program) files, while the existing filters for *Microsoft Visio*, *Publisher* and *Works* spreadsheets have seen lots of work. The OOXML filters have been tuned for greater fidelity to the original documents, and it's now possible to connect *LibreOffice* directly to *SharePoint* installations – a boon for enterprise users.

Another new feature is master document templates. Previously, master documents could be used to group a number of documents together, such as chapters in a book. It's now possible to create a template from a master document based on some initial content, so you can re-use it to create other master documents with that content in the future.

Outside of these major changes, there have also been hundreds of small tweaks across the whole suite. The *LibreOffice* team has spent much of the last five years cleaning up and refactoring old code – doing a lot of work under the hood to make the suite faster and easier to build – and while that was an admirable job, we're happy to see more user-facing changes in this version. A brilliant future awaits.



It's now possible to update and edit styles directly from the drop-down list.


lv013_libreoffice_rev.txt - LibreOffice Writer
File Edit View Insert Format Table Tools Window Help
Preformatted
Liberation Mono
10
Untitled 1 - LibreOffice Writer
File Edit View Insert Format Table Tools Window Help
Subtitle
DejaVu Sans
15

This is LibreOffice 4.3.3.2, as supplied with Xubuntu 14.10.

And this is LibreOffice 4.4 running on Arch. The icon theme is different, but you can see how many toolbar buttons have been removed and added.

A comparison of *Writer* toolbars between the current and previous release – and note the new sidebar on the right.

LINUX VOICE VERDICT

The GUI changes are very welcome – they make the suite more accessible without baffling existing users.

★★★★½

Icaros Desktop 2.0.3

The Amiga lives! Well, sort of. **Mike Saunders** explores this distribution of AROS, an Amiga-ish open source operating system.

Many of us at Linux Voice cut our teeth on the Commodore Amiga, arguably the best home computer of the late 80s and early 90s. When most PCs only had a buzzing piezo speaker for sound and struggled to display more than 16 colours, the Amiga's custom chipsets provided fantastic audio and graphics. And then you had a multi-tasking, lightning fast GUI operating system, a wealth of games and productivity tools, and a passionate community. Those were great days.

Still, Commodore was utterly abysmal at marketing and made a bunch of inexcusable errors. *Dangerous Streets*, for instance, was a detestably bad *Street Fighter II* clone that received a 3% score in *Amiga Power* – yet Commodore bundled it as the main game with its CD32 console. In short, the Amiga was awesome and everyone at the time ended up hating Commodore for being so rubbish.

Today, we have an open source AmigaOS-like operating system in the form of AROS (**www.aros.org**). It's small fry compared to other FOSS platforms like Linux and FreeBSD, but it has an active development community and keeps the spirit of the 90s alive. AROS implements many AmigaOS 3.1 APIs, and the x86 PC version includes a Motorola 68k CPU emulator, making it possible to run some games and programs that were written for the original machines.

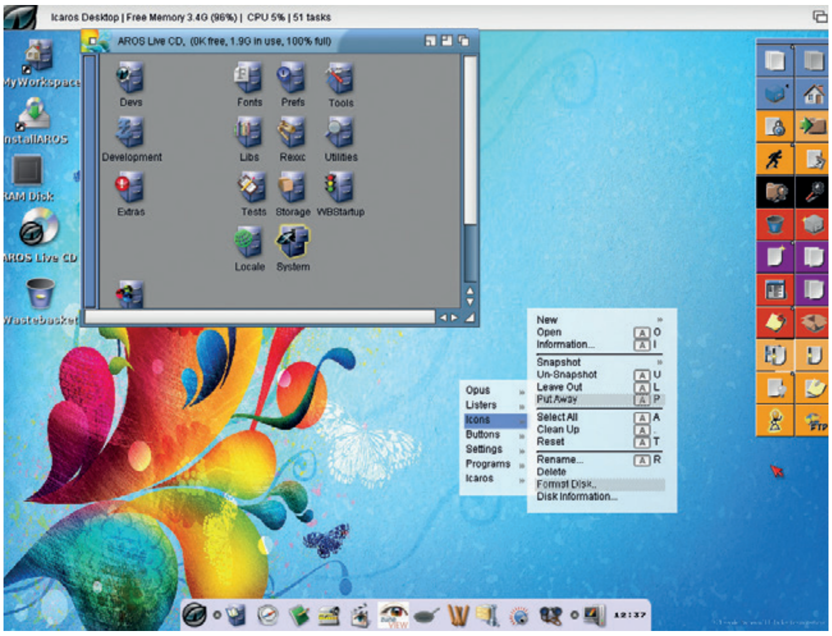
Icaros Desktop is a distribution of AROS, taking the core OS and adding various programs and desktop tweaks on top. You can burn it to a DVD and boot it on a real machine, and then install it to your hard drive – or test it out in a PC emulator such as *VirtualBox*.

Down memory lane

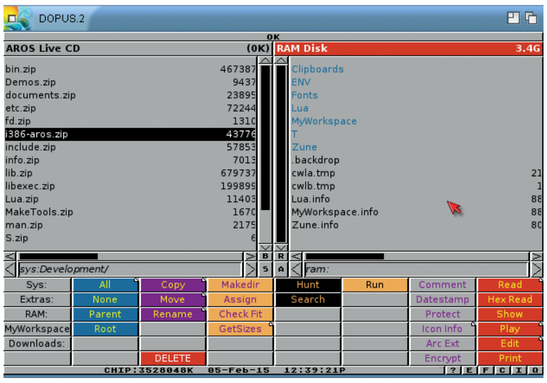
If you've used Amiga Workbench before, AROS will be immediately familiar: it's prettier and more colourful, but largely the same, with utilities and filesystem locations that have barely changed since the glory days. If you never had the opportunity to play around with an Amiga (and our hearts go out to you in that case), you'll find that it's not a difficult OS to explore. Double-clicking the AROS live CD icon on the desktop opens a file manager; this provides access to the utilities and demos included with the Icaros distribution. There's even a command line shell, although it's very different to those in Linux/Unix.

Icaros is supplied with a small collection of desktop programs: a web browser, a media player, a simple word processor and others. These applications are rather limited compared to the likes of *Firefox* and *LibreOffice*, as expected, but they run at a blistering pace. Of course, there's plenty more Amiga software out there, some of which is open source and still being updated – see **http://archives.aros-exec.org**.

AROS isn't going to replace your favourite desktop Linux distro any time soon, and it's very lacking in places. But it's a fascinating project, and we want to give it more recognition. As much as we feel at home in the comfy combination of GNU and Linux, we're always intrigued by other OS projects with different designs and goals. Maybe there are some things Linux developers can learn from AROS, or indeed other OS-recreation projects like Haiku OS... LV



Icaros has the basic Amiga Workbench layout, and spruces it up with extra panels and utilities.



```
DOPUS.2
                                   OK
AROS Live CD                 (0K) RAM Disk                          3.4G
bin.zip                    467387  Clipboards
Demos.zip                    9437  ENV
documents.zip               23895  Fonts
etc.zip                     72244  Lua
fd.zip                       1310  MyWorkspace
i386-aros.zip               43776  T
include.zip                 57853  Zune
info.zip                     7013  .backdrop
lib.zip                    679737  cwla.tmp                            21
libexec.zip                199899  cwlb.tmp                             1
Lua.zip                     11403  Lua.info                            88
MakeTools.zip                1670  MyWorkspace.info                    88
man.zip                      2175  Zune.info                           80
S.zip                           6
sys:Development/                   ram:
CHIP:3528048K   05-Feb-15   12:39:21P
```

Amiga fans will remember *Directory Opus* – it was open sourced in 2000 and runs on AROS/Icaros!

DATA

Web
http://vmwaros.blogspot.it
Developer
Icaros and AROS teams
Price
Free (open source licences)

LINUX VOICE VERDICT

Everything that we loved on the Amiga, running on modern hardware. Just don't expect loads of features.

★★★☆☆

CubieBoard A20

Looking for a little more power than the Raspberry Pi offers, **John Lane** wonders if the CubieBoard A20 can deliver...

DATA

Web
cubieboard.org
Price
£45.95

SPECIFICATIONS

CPU Mali400mp2, OpenGL ES GPU
RAM 1GB DDR3 @480MHz
Video out HDMI 1080p
Networking 10/100 Ethernet
Storage 4GB NAND flash
Connectors 2 x USB Host, 1 x micro SD slot, 1 x SATA, 1 x infrared, 2 x 48-pin headers exposing I2C, SPI, RGB/LVDS, CSI/TS, FM-IN, ADC, CVBS, VGA, SPDIF-OUT, R-TP

The Raspberry Pi is great, as you'll know if you've not ben stuck under a rock for the past three years. But there's so much more out there. You can now choose from loads of alternative small-board computers, including many that are packed with additional features for not much more money than the Pi. One such board, the CubieBoard, was launched in 2012 with native Ethernet, a built-in 4GB NAND flash drive and an on-board SATA connector. An upgrade in 2013 giving it a dual-core CPU and 1GB of memory was released as the CubieBoard 2, sometimes referred to as the CubieBoard A20 after the AllWinner A20 chip at its core. Although roughly twice the price of a Pi, it's still affordable, and it can run Linux and Android.

For your £45 you get the power, SATA and USB cables that you need to get started. You just need to add a power supply (a USB phone charger will do), keyboard, mouse and a HDMI display.

The underside of the board sports two 48-pin connectors among a heap of connection options. One such is VGA video – you can directly wire up a VGA port and use any old VGA monitor, opening up a the possibility of using it with hardware that may otherwise be thrown away, which could potentially be a huge factor when assessing total cost of ownership. Other nice touches include a reset button, audio line-in and an infrared port that will come in handy if you want to use a remote control with it.

"Those comfortable with a more bare-bones approach may prefer to use Arch Linux ARM."

There is a micro-SD card slot, which is one of several places the board will boot from – you can also boot from the NAND flash ROM or via the network but you can't boot from the on-board SATA interface. There is also a variant of the board that has a second micro-SD card slot instead of the NAND flash ROM.

An additional mini-USB On The Go port is provided to connect the CubieBoard to a desktop PC to perform flash updates using AllWinner's *LiveSuit* software. This mode of operation is called FEL mode and operates similarly to the recovery mode available on some smartphones.

The on-board NAND flash ROM comes pre-loaded with Android, so you can plug in and use the device straight away. But you'll most likely want to install a Linux distro to use it for your own projects, and this is where the fun begins...

It's good to keep talking

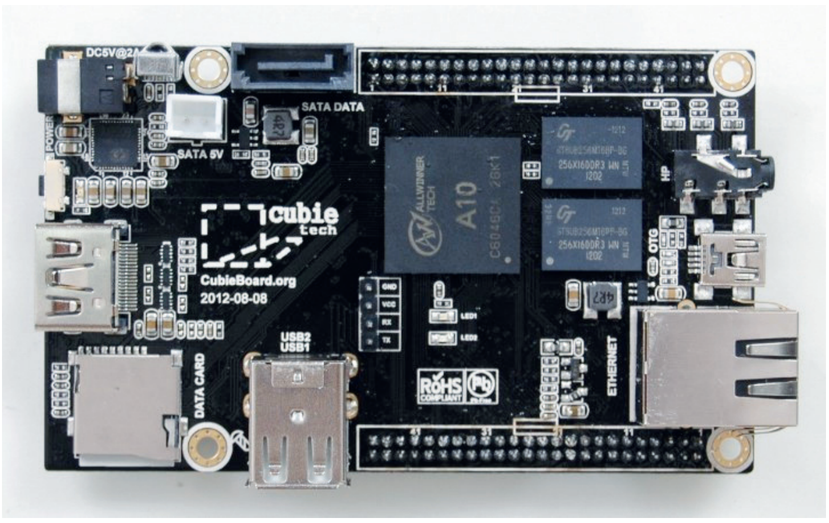
Something to bear in mind is that, unlike the Raspberry Pi, the primary way of communicating with the CubieBoard is over its serial port. You can activate its HDMI port during boot, but a serial port is handy if you want to see early boot messages, something we found essential for troubleshooting. We used a USB serial adapter and the GNU *screen* tool for this. To install another operating system you'll need an SD card; the easiest way to get started is to download a prepared image for your chosen Linux distro. For those coming from a Pi background, the obvious choice is a distro called Cubian, a Debian spin inspired by Raspbian. For a more hardcore experience, "ArchLinux ARM" supports it too.

Cubian comes in three flavours, a desktop version with the Mate desktop or a text mode called Nano, either with or without the HDMI display support; the latter leaves more memory free for you to use. It's a 2GB image that will require a micro SD card of at least that size. The download is a 7-Zip archive so you'll need to install the appropriate tool to unpack it (try **apt-get install p7zip-full** or **pacman -S p7zip**).

Install your chosen image onto a spare micro SD card. Do this from another machine with **dd** (take the usual precautions to ensure you write to the correct device). Insert the card into the CubieBoard, attach a keyboard and monitor and start it up.

You're given the chance to change some options when you start Cubian for the first time, and this is a good time to localise your install and set a password. You can then log in and you'll find the usual gamut of applications. The included *SMPlayer* gave good results streaming full-screen standard video across the

The CubieBoard 2 is marginally larger than the Raspberry Pi but offers a lot more connection options.



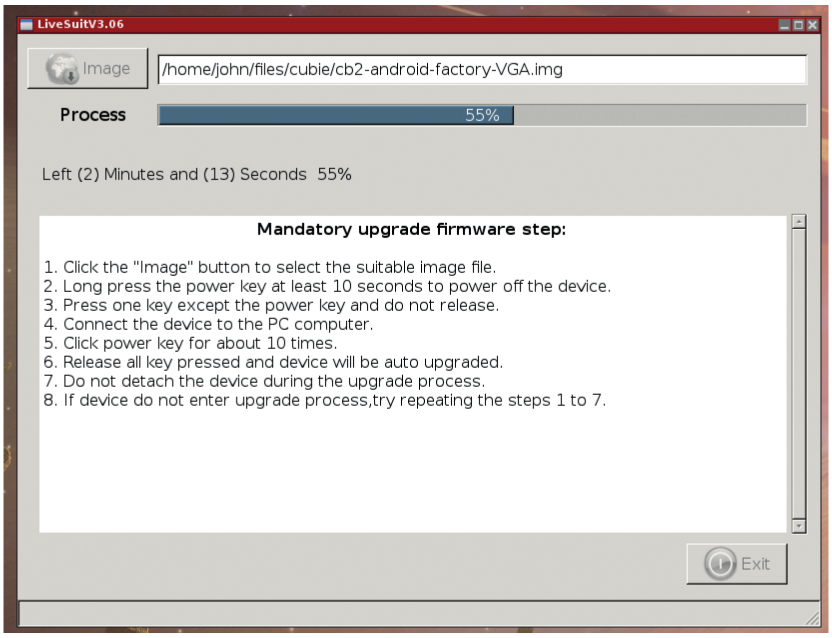
network, but the device struggled with high definition material, even after installing the more capable *VLC*.

Cubian includes a tool to transfer your installation from an SD card onto the internal NAND ROM. All you need to do is install and run the **cubian-nandinstall** package, shut down, remove the SD card and then re-start. However, this failed for us, and attempts to gain answers via the forums were unfruitful.

Those comfortable with a more bare-bones approach may prefer Arch Linux ARM. If you follow the CubieBoard 2 install instructions to the letter, you'll be able to boot the board without any problems and log in to Arch Linux over SSH. The default Arch install leaves you on your own, however – you'll have to manually configure the HDMI output unless you're happy with the serial port and/or SSH. However, the active forum is a haven when in need of support.

The inclusion of a powered SATA port sets it apart from most of the other small-board computers available, while having twice as many cores and memory affords more acceptable performance.

The CubieBoard and Linux-sunxi forums are good but there is nothing like the ecosystem that has grown up around the Pi. However, this has the greater impact on beginners, and we feel that the CubieBoard isn't aimed at more advanced users. Experts can use Arch Linux ARM and its forum, where there is a dedicated AllWinner sub-forum.



```
/home/john/files/cubie/cb2-android-factory-VGA.img

Left (2) Minutes and (13) Seconds  55%

Mandatory upgrade firmware step:

1. Click the "Image" button to select the suitable image file.
2. Long press the power key at least 10 seconds to power off the device.
3. Press one key except the power key and do not release.
4. Connect the device to the PC computer.
5. Click power key for about 10 times.
6. Release all key pressed and device will be auto upgraded.
7. Do not detach the device during the upgrade process.
8. If device do not enter upgrade process,try repeating the steps 1 to 7.
```

Use the *LiveSuit* installer to upload images to the NAND Flash ROM.

LINUX VOICE VERDICT

A well-capable board with varied connectivity options and good support for Linux and Android.

★★★★☆

BUY LINUXVOICE MUGS AND T-SHIRTS!

LINUXVOICE
FREE SOFTWARE | FREE SPEECH
"IT'S EASIER TO ASK FORGIVENESS THAN IT IS TO GET PERMISSION"
Rear Admiral Dr Grace Murray Hopper

shop.linuxvoice.com

Clipping Through

Ben Everard saves £1,000 by reading about San Francisco instead of going there.

Clipping Through is a travel book about a computer games conference. We'll be honest, that sounds like a strange premise for a book. However, like many good travel books, it's not really about the place – the Games Developer Conference is just the backdrop to Leigh Alexander's treaty on the state of the video games industry. She interweaves tales from her long journalistic career in video games with a warts-and-all diary of the conference.

Actually, it's less of a diary and more a series of snapshots. These snapshots include interviews, discussions on interpretations of games, and the inequality evident in San Fransisco. It's well written, and surprisingly coherent given the wide range of subjects covered.

This book is a view of all the aspects of the video games industry that don't get covered elsewhere: The odds-and-ends, the ugly bits, the alcohol-fuelled bits. It's a book for people who want to know what goes on behind the scenes of games, but not the conventional behind-the-scenes coverage.

There's no print version, but for $5 (with the option to pay more), you get the book in a range of digital formats (Epub, MOBI, PDF and DOCX). We have serious concerns about anyone who wishes to read a book in DOCX format. Oh, and before you ask, no, this isn't a book about ethics in video games.

LINUX VOICE VERDICT

Author Leigh Alexander
Publisher Gum Road
ISBN None
Price $5+

Surprisingly readable for its breadth of subjects, and very evocative of a place and time.

★★★★☆

Liz Ryerson, the artist featured on the cover, also wrote the afterword sharing her experiences in the industry.

Free Culture: The nature and future of creativity

You'll never take **Ben Everard**'s freedom.

There are a few books that we would consider essential reading for anyone interested in open source, and *Free Culture* by Larry Lessig is one of them. It's a decade old, but age hasn't lessened its importance. Perhaps the last 10 years have actually lead to the message of the book being more acute now. As you can probably tell from the title, it's not about software specifically, but about the ability to share any creative works. This should come as no surprise, since Lessig is one of the authors of the Creative Commons licences.

Lessig is a lawyer by training, and that inevitably means the book is heavily influenced by the legalities and legal challenges around sharing, particularly in America. This may not sound like a great read, but let us assure you that it is. Lessig's passion flows through every word in his highly thought-out defence of the right to share. His primary argument is against the continued lengthening of copyright terms at a rate which appears to make them infinite. For anyone who is unsure which side of the debate they sit on, Lessig's arguments in favour of freedom are compelling, at least to this reviewer. *Free Culture* puts the fight for freedom in its historical context (it goes back to Shakespeare) and highlights the forces that are working against it. It shows the weapons and tactics that the enemies of freedom are using to bring all content into their protected domain.

LINUX VOICE VERDICT

Author Lawrence Lessig
Publisher Penguin Books
ISBN 978-0143034650
Price £10.56

Probably the most interesting book ever written on the subject of copyright law.

★★★★★

FREE CULTURE
HOW BIG MEDIA USES TECHNOLOGY AND THE LAW TO LOCK DOWN CULTURE AND CONTROL CREATIVITY
LAWRENCE LESSIG

We use Creative Common's licences – co-written by Lessig – when we free our issues because we believe in Free Culture.

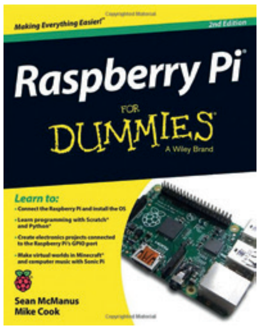
Raspberry Pi for Dummies (2nd Edition)

Graham Morrison finds the perfect book for David Cameron.

Despite the launch of a new Raspberry Pi model, and the likely glut of new book editions that are going to flood the market, there's nothing in the new model that breaks compatibility with the old, which means books like this will remain just as useful in the quad-core era of Raspberry Pi 2.0.

This is just as well, because this Dummies title is an excellent introduction for the complete beginner to both the hardware and Linux itself. A lot of pages are dedicated to setting up, for example, discussing what should be plugged in and where. There are chapters on using the desktop, and an introduction to the shell.

We also like the creativity of the sections dealing with Scratch programming, followed quite naturally with some more advanced ideas written in Python. The book is topped off with a substantial section on building your own circuits for more ambitious projects. It's exactly this kind of progression that the Raspberry Pi was created to nurture, and it's good to see a well written book from such a successful brand taking the trouble to get this right.


Mike Cook used to write a column for *Micro User* in the 80s .

LINUX VOICE VERDICT

Author Sean McManus & Mike Cook
Publisher Wiley
ISBN 978-1-118-90491-6
Price £17.99

A comprehensive and gentle overview of everything you need to get started.

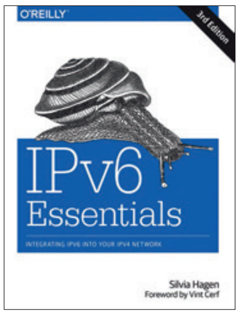
★★★★☆

IPv6 Essentials (3rd edition)

Pack this for the IPv4 apocalypse, says **Graham Morrison**

We've been told for a while that IPv4 is untenable for the future of the internet. It's going to be impossible to have a shiny new Internet of Things, with its conferences in California and ear sensors, with IPv4's pathetic 4.3 billion address limitation. But like ADSL, it keeps hanging on and adapting to whatever our crazy world of computers can throw at it.

IPv6 is the answer, and it's finally becoming more prevalent. Many ISPs now provide compatible routers, even if they've not turned the facility on, and Google's IPv6 adoption statistics have started to ramp up – currently at 6% of internet traffic. This is where this book comes in. It's not too long, and while its style is CompSci rather than entertaining, we liked the practical examples that used *WireShark* to monitor IPv6 DNS traffic, even when most chapters finish with a huge list of RFCs for reference. But the best thing about this book is that it's relatively easy to understand. That's not a simple trick to pull off with network concepts, and it's the reason why we'd recommend this book as any reader's first step to IPv6 migration.


The end of IPv4 is nigh.

LINUX VOICE VERDICT

Author Silvia Hagen
Publisher O'Reilly
ISBN 978-1-449-31921-2
Price 26.50

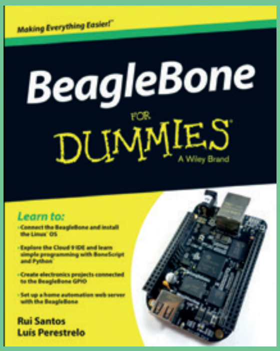
A great overview that leaves you knowing what you need to do next.

★★★☆☆

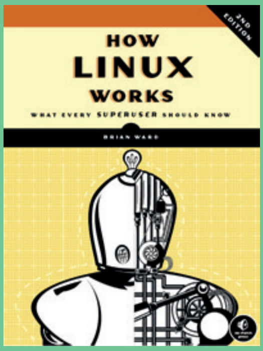
ALSO RELEASED...

Kathy Sierra is back and she's writing about users!

Badass: Making Users Awesome
It's fantastic to see Kathy Sierra back. We've loved many of her Head First titles, and she has a wonderful knack of writing insightful things about users. This is a book about creating successful users, and we can't wait to read it.


Be the cool kid in class with a Beaglebone.

BeagleBone for Dummies
After enjoying *Raspberry Pi for Dummies* (see left), we're now looking forward to the release of this – hopefully the same treatment but for the more open and equally cool BeagleBone hardware. Arguably, a book like this is more important as there's less help available online.


Shouldn't that be GNU/Linux?

How Linux Works
This is the second edition of a book we already think is good. It's ideal for those users who want to take their Linux knowledge further, and is perfect if you've been messing around with the desktop and command line for a while and want to know what makes it all tick.

LINUX VOICE

BSD DISTROS

GROUP TEST

There's more to open source than Linux. **Mayank Sharma** takes an excursion to test some BSD distros.

On Test

FreeBSD

URL www.freebsd.org
VERSION 10.1
LICENCE Simplified BSD
Can the most prominent daemon cast a spell on the rest?

OpenBSD

URL www.openbsd.org
VERSION 5.6
LICENCE ISC
Unflashy OS with an unrivalled focus on security.

NetBSD

URL www.netbsd.org
VERSION 6.1.5
LICENCE Simplified BSD
One of the oldest BSDs in existence.

DragonFly BSD

URL www.dragonflybsd.org
VERSION 4.0.2
LICENCE Modified BSD
Has this FreeBSD fork been able to carve a user-base?

GhostBSD

URL www.ghostbsd.org
VERSION 4.0
LICENCE BSD
Does this BSD designed for the desktop offer anything of note?

PC-BSD

URL www.pcbsd.org
VERSION 10.1
LICENCE BSD
Bucks theBSD convention with a focus on usability.

BSD Distros

For the distro hoppers among us, a Linux distribution is just a collection of applications and utilities. We can be productive with any distro as long as it gives us access to our cherished and trusted tools. So how about a diversion into the land of the BSDs?

While they haven't caught the fancy of the mainstream tech press, the BSDs are known for their robustness, reliability and security and are fairly popular with system administrators. That said, you can slap popular open source apps on top and use BSDs for everyday desktop computing tasks, such as browsing the web, listening to music, watching DVDs, playing games and reading PDFs. Also of note is their devoted community of developers and users.

The modern day BSDs that we have on test in this feature can be traced back to the 1970s. BSD stands for the Berkeley Software Distribution. It was the name of the toolkit of enhancements for UNIX that was created at the University of California, Berkeley. In contrast to UNIX, which was developed at Bell Labs, BSD was created by students and faculty at the University. BSD was distributed as a package of software enhancements for UNIX that made it useful in the real world, outside of a research laboratory. Over time, BSD evolved and replaced every part of UNIX and became a usable operating system in itself. The current stable of BSD distros are a family of OSes that are derived from the original.

The three most notable descendants in current use are FreeBSD, OpenBSD and NetBSD. The majority of current BSD OSes are open source and available for download for free under the BSD Licence, with Mac's OS X being the most notable exception. In this group test we'll evaluate the strengths of the most popular BSD distros and help you pick one that's both easy to use and can be used for a wide variety of applications.

"The modern day BSDs that we have here can be traced back to the 1970s."

BSD vs Linux

Both Linux and the BSDs are free and open source, Unix-like operating systems, and use much of the same software. So what sets them apart from each other? For starters, Linux and BSD have a different lineage. Technically speaking, Linux is just a kernel. To produce a usable OS, each distro must glue a bunch of software on top of the kernel. On the other hand, a BSD project maintains the entire OS and not only the kernel. Another significant difference is the licensing. Linux uses the GPL, which requires that developers release the source code for their modifications. BSDs, in contrast use the BSD Licence which allows modifications to be kept under wraps if the developer so requires.

Popular products based on BSD

Daemons at work.

There are many open source as well as commercial products based on BSD, due to its technical prowess and permissive licensing. Popular open source products include NAS distros like FreeNAS and NAS4Free and firewall projects such as the embedded Monowall distro and its fork for regular computers, PFSense.

Then there's the open source Darwin project by Apple, which produces the core components upon which the company's proprietary OSes, OS X and iOS, are based. Darwin integrates a customised version of BSD. BSD is also used as the basis for the filesystems and networking of OS X.

There are other commercial products based on BSD by multinational hardware and software companies, such as Dell's iSCSI SAN arrays. Silicon Graphics International also uses FreeBSD in its ArcFiniti MAID (Massive Array of Idle Drives) disk arrays and so does Sony in its PlayStation 3, PlayStation 4 and PlayStation Vita gaming consoles.

FreeBSD seems to be the most popular of the BSDs that is found powering both open source and proprietary products. Juniper Network's JUNOS is based on FreeBSD, and is also used in the Netflix Open Connect appliance. FreeBSD also powers the popular messaging app, WhatsApp, and the project received $1 million recently from WhatsApp's co-founder.

OpenBSD

Is it open for all?

OpenBSD is regarded as one of the safest operating systems. Thanks to its rigorous code audit and security-first development model, it's a popular option for security-conscious applications such as firewalls, intrusion-detection systems, and general-purpose servers. OpenBSD is designed for experienced users who know what they are getting themselves into and are willing to tussle with its peculiarities in order to get the job done. So although it doesn't ship as a desktop out of the box, you can configure it as one if you're willing to spend time getting used to it.

To get the distro on your machine you have to labour through its text-based installer. It's well laid-out and presents each option with ample information, but does require a competent and knowledgeable operator. That said, the default options are sensible and it even suggests an auto-partitioning scheme, which is a definite aid for new users.

OpenBSD's sole purpose is to be the most secure OS, which usually puts it at odds with convenience.

Build from scratch

Also, since it's designed for specialised usage, OpenBSD's hardware support focuses more on enterprise-grade and even virtual hardware. Still, unless you've got some exotic top-of-the line hardware or a brand-new device, you should be able to get it to work with OpenBSD. The developers keep adding drivers and have recently added a new touchpad driver that supports Broadcom multi-touch trackpads found on newer Apple MacBook, MacBook Pro, and MacBook Air laptops. OpenBSD now also supports AMD Radeon graphics cards thanks to code ported from FreeBSD.

After installation you get a fairly barebones (but incredibly fast) system with a handful of services enabled, and a minimal desktop running the FVWM window manager, if you've selected it during the installation. This base has all the tools you need to transform it into a dedicated server or a desktop.

The latest version of OpenBSD includes popular desktop software including KDE 4.13, Gnome 3.12, Xfce 4.10, *Firefox 31*, *LibreOffice 4.1.6* and others. As expected, the thorough security vetting means that some of these are a couple of versions behind their latest iterations. OpenBSD is also popular for its documentation and you can find loads of information on the web that'll help you transform this bare-bones OpenBSD install into a full-fledged desktop. Depending on how fast your internet connection lets you fetch packages from OpenBSD's mirrors, you can set up a desktop in under an hour.

But all said and done, OpenBSD isn't designed to be a comfortable desktop distro. It's certainly possible to use it as desktop platform, but that isn't a top priority for its developers. Unlike regular desktop distros, OpenBSD isn't meant to make decisions on your behalf or make your life easier with automated routines. Rather, OpenBSD is designed to be secure and reliable, and this is something it's very adept at. The fact that you can use such a system as a regular desktop is an additional benefit.

VERDICT

A stable OS with a rock-solid foundation and a security-first approach.

★★★★☆

GhostBSD

I see desktop users!

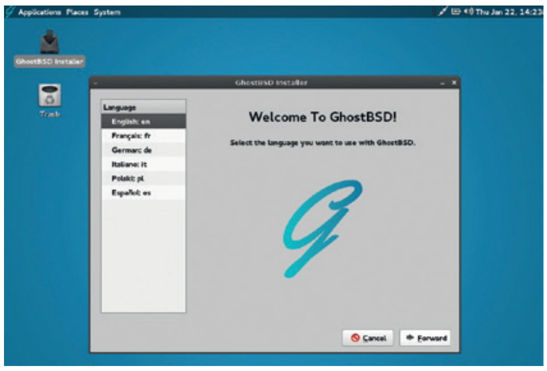
Here's a BSD that closely resembles a desktop Linux distro in form and function. GhostBSD is available for 32-bit and 64-bit machines both as ISO and IMG files for optical drives and flash disks respectively. Unlike most of the other BSDs on test here, GhostBSD boots straight into a live graphical desktop environment. Earlier versions of the OS shipped multiple desktops, but the current release is available with only the lightweight Mate desktop.

On the desktop, the live environment offers three different layouts. There's the classic look, with the menu on top and a panel at the bottom; a minimalist view; and a layout with a simple dock at the bottom. Since GhostBSD is based on FreeBSD, it allows users to access FreeBSD's repository of packages via its new command-line package manager, **pkgng**. However, GhostBSD doesn't have a package management system of its own and also lacks a graphical app for installing packages, so like most others BSDs you can't escape the command-line.

That said, in contrast to most other BSDs, GhostBSD includes a multilingual graphical installer, which offers an automated partitioning scheme if you wish GhostBSD to take over the entire disk. Unfortunately, the installer is pretty buggy and crashed several times without spitting any error messages.

Also, GhostBSD includes older versions of some apps, such as its Mate desktop. It includes Mate 1.6 while the latest version is 1.8. This version is available in its repositories, but again, upgrading it is easier for someone familiar with FreeBSD's **pkgng** utility. GhostBSD is a good attempt at shipping a FreeBSD-based OS with the convenience of a familiar desktop. It's lightweight and functional, but doesn't really help new users escape the clutches of the command-line utilities.

"GhostBSD boots straight into a live graphical desktop environment."



Welcome To GhostBSD!

Select the language you want to use with GhostBSD.

While the current release includes the Mate desktop, the developer is looking at alternatives for future releases.

VERDICT
Lightweight BSD loaded with apps that ships as a live desktop.
★★★☆☆

DragonFly BSD

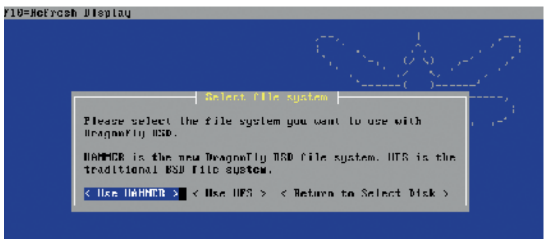
Hovering on the desktop.

DragonFly BSD is a popular fork of FreeBSD that is now developed in a direction of its own and is considered one of the main BSD distros. The OS has diverged significantly from FreeBSD and is popular for its implementation of virtual kernels and a feature-rich 64-bit filesystem called HAMMER, which has built-in mirroring, instant crash recovery, and historic access functionality. It's popular for its Sun ZFS-like features with a friendlier licence. However, since HAMMER doesn't perform well on drives smaller than 50GB, DragonFly BSD's installer also lets you use the BSD standard UFS filesystem.

The installation medium for the OS is available for 64-bit architectures only. The project releases an ISO image for optical drives and an IMG file for installing via USB. Both boot into a live environment and let you log in as root and check the compatibility of your hardware from the command line. Once you're satisfied you can fire up the installer. DragonFly BSD also has a menu-driven text-based installer that'll help you set up a single OS installation in a matter of minutes. There's also a configuration tool that you must run through at the end of the install to set up key aspects of the system such as the networking options.

Like most BSDs, DragonFly doesn't install a graphical desktop, but don't underestimate its graphical prowess. The latest version of the OS supports GPUs from the Haswell family, and OpenGL acceleration is available out of the box on supported i915 and Radeon GPUs. There's also a touchscreen driver for the Acer C720P notebook.

To set up a desktop environment like Xfce, KDE, Gnome or Mate, you'll have to labour through the documentation available on the project's website. Again, the saving grace is its support for binaries and third-party apps. DragonFly BSD uses FreeBSD Ports as a base for its own ports collection (called Delta Ports), and you can also install packages using FreeBSD's **pkgng**. DragonFly BSD focuses more on the desktop users than some other BSDs, it still requires considerable dexterity on the command-line.



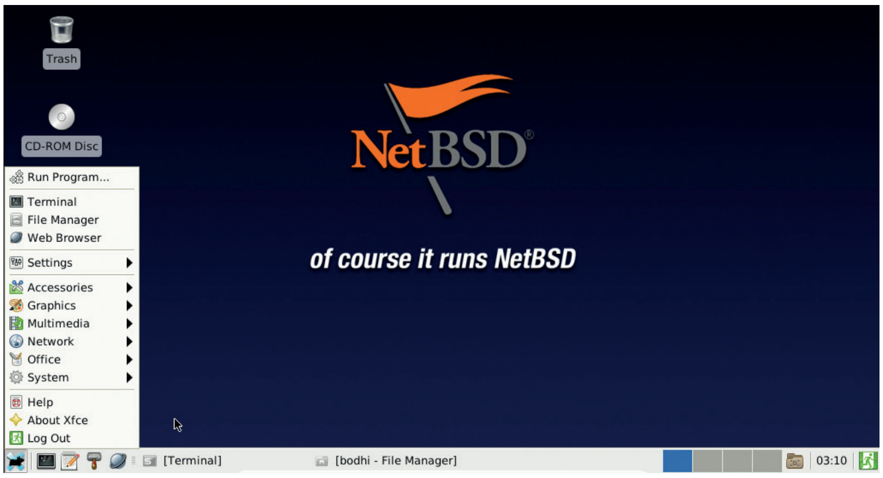
```
F10=Refresh Display

            Select file system

Please select the file system you want to use with
DragonFly BSD.

HAMMER is the new DragonFly BSD file system. UFS is the
traditional BSD file system.

< Use HAMMER >  < Use UFS >  < Return to Select Disk >
```

DragonFly BSD was started by Matt Dillon, who had earlier developed the *Dice* C compiler for the Amiga.

VERDICT
Though it doesn't ignore desktop users, it's best for stable server deployments.
★★★☆☆

NetBSD

The Universal BSD.



NetBSD's rock-solid foundation and portability were a big draw for NASA, which used it for a project on the International Space Station.

True to its tagline, which reads "Of course it runs NetBSD", this BSD distro that's been around since the early 90s is very portable and runs on over 50 different hardware platforms across 15 processor architectures.

Another major feature of the OS is that unlike typical desktop Linux releases that come out about every six months, new NetBSD releases are fairly infrequent. This release schedule is preferred by its primary users, who wouldn't want to pull down their servers for upgrades often. However, you aren't necessarily stuck with outdated software, as its package management system, **pkgsrc**, tracks the latest version of upstream software. While it does give access to all kinds of software, the fact that it lacks a graphical package management app is rather limiting.

NetBSD uses an *ncurses*-based menu-driven installer, which is pretty simple to navigate and besides the partition management steps, shouldn't pose any issues to experienced distro-hoppers. At the end of the installation, you'll be asked to configure some essential aspects. Also, while the Full installation scheme installs the basic X window system components, it doesn't include a graphical desktop.

Finger flexing

The default installation is pretty minimal. You'll have to set up the package manager and then install the required packages. This isn't much of a task thanks to NetBSD's binary package, **pkgin**, using which you can install popular open source desktops and apps. A couple of years back there was news of a light-desktop initiative to quickly turn a NetBSD installation into a lightweight desktop based on LXDE and modelled after Lubuntu, but it seems to have died.

Furthermore, if you're spoilt by graphical configuration tools on modern Linux distros, you wouldn't get very far with NetBSD as it requires everything to be set up by hand. Editing files under **/etc** to get the wireless card to connect bought back bittersweet memories. You'll have to add users from the command line and also create mount points and mount optical discs manually. You'll also have to get familiar with NetBSD's use of the **rc.d** system to control services, which is similar to System V but without runlevels. In fact, you wouldn't get very far with NetBSD without having first read through the very detailed NetBSD Guide. The project has extensive documentation and a detailed wiki that'll help you get familiar with its peculiarities. NetBSD isn't the easiest of OSes to get started with, but its vanilla approach makes it ideal for security-conscious users who like to be in charge of each and every component running inside their computer.

VERDICT

The open source distro that can run on everything from a toaster to the space station.

★★★☆☆

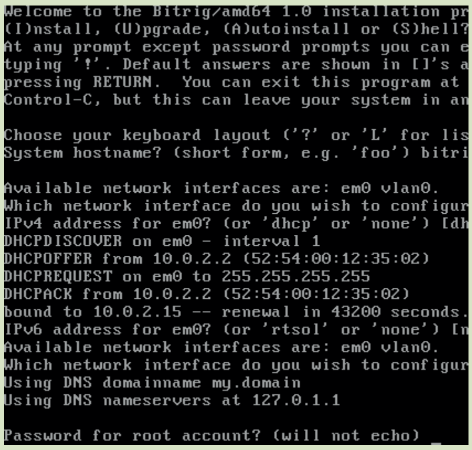
BSD-based distros

Wait! There's more?

There's a long list of BSD-based OSes that are under active development. ArchBSD is a lightweight operating system that aims to club the flexibility and philosophy of Arch Linux with BSD-based operating systems. Then there's Debian GNU/kFreeBSD, which swaps out the Linux kernel and instead uses the FreeBSD kernel together with GNU-based userland utilities and glibc. It's developed by the Debian project, which maintains two ports based on the FreeBSD kernel, 'kfreebsd-i386' and 'kfreebsd-amd64'. Similarly, there's Gentoo/FreeBSD, which is a subproject to port unique Gentoo features such as the *Portage* package management system to FreeBSD. There's also the Evoke project, which produces a small live FreeBSD environment geared toward developers and system administrators.

FreeBSD is also the basis for the popular Monowall embedded firewall distro that provides a small image that users can put on CF cards. While you can use Monowall on a generic PC, the PFSense project produces a firewall designed for computers.

MidnightBSD is a FreeBSD fork that aims to create a desktop-friendly BSD operating system and has also forked FreeBSD's *ports* package management system to create its own called *mports*. There are several NetBSD-based projects as well, such as the Jibbed bootable live CD. Another is BlackBSD, which is a live CD that includes a bunch of security tools such as *Nmap*, *Nessus*, *Snort*, *Rapid7* and others.



```
Welcome to the Bitrig/amd64 1.0 installation pr
(I)nstall, (U)pgrade, (A)utoinstall or (S)hell?
At any prompt except password prompts you can e
typing '!'. Default answers are shown in []'s a
pressing RETURN.  You can exit this program at
Control-C, but this can leave your system in an

Choose your keyboard layout ('?' or 'L' for lis
System hostname? (short form, e.g. 'foo') bitri

Available network interfaces are: em0 vlan0.
Which network interface do you wish to configur
IPv4 address for em0? (or 'dhcp' or 'none') [dh
DHCPDISCOVER on em0 - interval 1
DHCPOFFER from 10.0.2.2 (52:54:00:12:35:02)
DHCPREQUEST on em0 to 255.255.255.255
DHCPACK from 10.0.2.2 (52:54:00:12:35:02)
bound to 10.0.2.15 -- renewal in 43200 seconds.
IPv6 address for em0? (or 'rtsol' or 'none') [n
Available network interfaces are: em0 vlan0.
Which network interface do you wish to configur
Using DNS domainname my.domain
Using DNS nameservers at 127.0.1.1

Password for root account? (will not echo) _
```

Bitrig is an OpenBSD fork that aims to be more modern in development and support than OpenBSD and has recently hit version 1.0.

FreeBSD vs PC-BSD

The Jedi and his prodigious padawan.

FreeBSD is the most used of the BSDs. The OS had its first release back in 1993. FreeBSD uses a text-based installer that is also used by several other derivatives as well. It's feature-rich and extensive with adequate defaults to aid first-time users. The installer hands off to a post-installation configuration screen from where you can set up important aspects of your computer such as the network interfaces.

FreeBSD doesn't install a graphical environment by default, but all the popular ones are available in the FreeBSD ports collection. Because of the ports collection, you can easily configure and use FreeBSD as a web server, mail server, or a firewall.

New users will have to follow the guides on the internet to turn their bare minimum FreeBSD install into something useful since there's nothing intuitive about the process. In addition to pulling in packages you'll also have to tweak configuration files. This might sound cumbersome, but is actually pretty straightforward and at the end produces a finely tuned aerodynamic system that does exactly what you want it to do and nothing else.

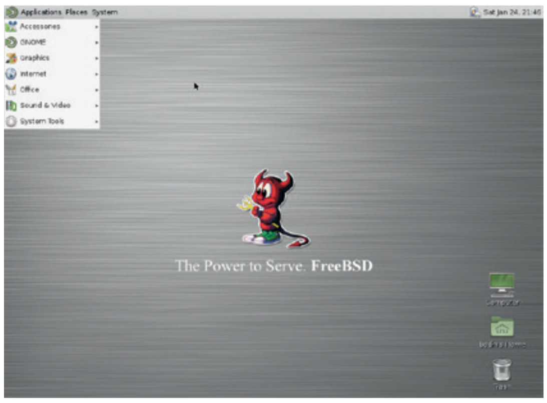
Angels and daemons

FreeBSD's popularity has spawned several derivatives. PC-BSD is one such descendant that's made a name for itself by extending FreeBSD's famed stability to everyday desktop users.

PC-BSD employs a graphical installer that's easy to navigate. It offers good defaults for new users as well as plenty of options for advanced users, especially in the disk partitioning step. By default the OS installs the KDE desktop on all machines that have more than 2GB of RAM and LXDE on those with less. The installer gives you access to full-featured desktops such as Gnome and Cinnamon as well as lightweight alternatives such as Xfce, Mate and even minuscule ones like Openbox and IceWM. You can also install additional components such as drivers for Nvidia cards, OpenJDK, the *Chromium* web browser and a lot more.

Post-installation, PC-BSD takes you through a series of steps to set up your computer. The OS also includes several custom tools to ease management. Its Control Panel app helps you manage different aspects of your installation, such as adding new users, configuring network connections, setting up the firewall and more. Then there's the backup tool, called *Life Preserver*, which can sync to a remote FreeNAS system securely, using rsync and SSH.

PC-BSD also has all the popular open source desktop apps that you can find on a typical Linux desktop distro. The *Firefox* browser is equipped to play Flash content and it includes the *VLC* media player to handle files in proprietary formats. PC-BSD uses FreeBSD's ports and also publishes packages in its own push-button installer (PBI) file format, which you can install via its *AppCafe* graphical package manager. *AppCafe* is a very modern and user-friendly package manager. It displays latest releases, recommends apps and also lets you search for particular apps. The tool also lets you know when newer versions for installed apps are available.

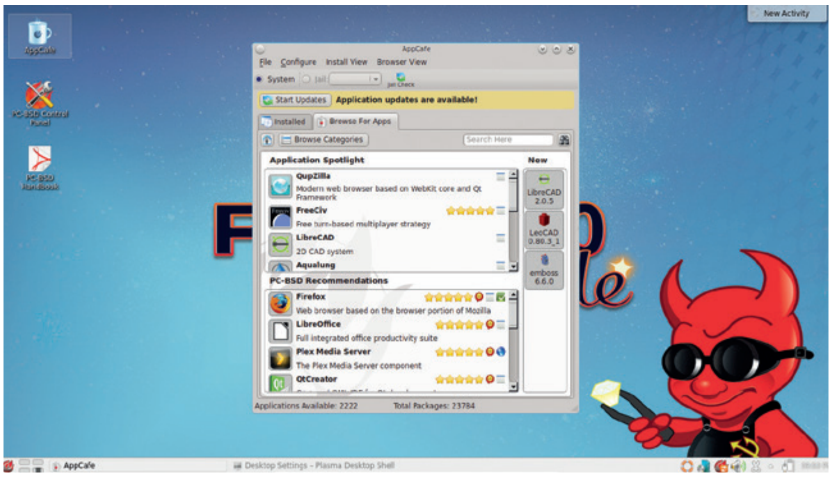


FreeBSD is all about DIY. Heck, you'll even have to install the package management tool before pulling in packages!

Furthermore, if you wish to deploy a BSD-based server instead of a desktop,

"PC-BSD has all the popular desktop apps that you'll find on a typical Linux distro."

you can use the PC-BSD installer to setup TrueOS. This is basically a vanilla FreeBSD installation with a bunch of extra PC-BSD packages for easier management. One such tool is *Warden*, which is used for creating isolated instances of the OS that can each run different services such as the *Apache* web server, or the *MySQL* database server. These isolated instances are known as jails and are one of the best examples of FreeBSD's technical superiority. PC-BSD's *Warden* tool lets you manage these jails both graphically and from the command line.



Application updates are available!

QupZilla — Modern web browser based on WebKit core and Qt Framework
FreeCiv — Free turn-based multiplayer strategy
LibreCAD — 2D CAD system
Aqualung
Firefox — Web browser based on the browser portion of Mozilla
LibreOffice — Full integrated office productivity suite
Plex Media Server — The Plex Media Server component
QtCreator

Applications Available: 2222 Total Packages: 23784

PC-BSD also lets you manage installed apps and jails from a remote machine.

VERDICT

FREEBSD The best enterprise-grade and proven BSD for your tuning pleasure.
★★★★☆

PC-BSD Deliver the same open source goodness with a different base.
★★★★★

OUR VERDICT

BSD distros

Despite them both being open source operating systems that share a lot of similar tools and utilities, finding a BSD replacement for Linux isn't a straightforward task. It takes more effort to get the BSDs set up as a desktop or even as a production-ready server when compared to most Linux distributions. The process is involved and manual: Mandriva and Ubuntu put a huge amount of work into making Linux more accessible in the mid-2000s, but a similar change hasn't happened yet in the BSD world.

While the BSDs all share a common ancestor, they have all diverged significantly over the years and have managed to create a niche for themselves. NetBSD is similar to FreeBSD in many ways, and the teams share developers and code. However, NetBSD's main purpose is to provide an OS that can be ported to any hardware platform. Then there's OpenBSD, which branched off from NetBSD with the goal of becoming the most secure BSD, even if that comes at the price of making it less user-friendly. Another popular FreeBSD-fork, DragonFly BSD, is popular for setting up virtual hosting environments on shared servers.

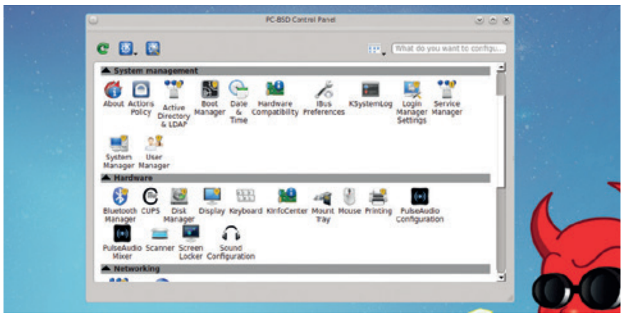
GhostBSD is a nice attempt, but at its current stage it's more of a teaser of what a FreeBSD-based lightweight desktop might look like.

FreeBSD has made a name for itself with its stability and is used by internet-oriented companies such as Yahoo and WhatsApp. If you're willing to spend some time with it, you can use FreeBSD as a full-featured and stable desktop or development workstation. Its *ports* package management system has eased software installation considerably and has been adopted by several other BSD projects as well. FreeBSD also trumps the other BSDs with its strong community and extensive documentation.

"Finding a BSD replacement for Linux isn't a straightforward task."

Double win

But if you're looking for point-and-click simplicity, there's no better bet than PC-BSD. This operating system is as close as BSD can get to Linux. It offers the same functionality, convenience and applications that Linux users are used to. In addition to its skills on the desktop, you can also use it to power your servers. PC-BSD proudly exposes its FreeBSD base and makes it accessible to new users by adding convenient tools for the administrators.



PC-BSD is backed by iXsystems, a company that has strong ties with FreeBSD and also sponsors the development of FreeNAS.

1st PC-BSD
Licence BSD **Version** 10.1

www.pcbsd.org
Drop-in replacement for Linux on the desktop and on the servers.

2nd FreeBSD
Licence Simplified BSD **Version** 10.1

www.freebsd.org
Put in a little time and effort to build a stable server or a functional desktop.

3rd GhostBSD
Licence BSD **Version** 4.0

www.ghostbsd.org
A usable lightweight desktop BSD.

4th OpenBSD
Licence ISC **Version** 5.6

www.openbsd.org
Best suited for security-centric implementations.

5th DragonFly BSD
Licence 4.0.2 **Version** Modified BSD

www.dragonflybsd.org
Its kernel enhancements make it ideal for stable server-centric deployments.

6th NetBSD
Licence Simplified BSD **Version** 6.1.5

www.netbsd.org
The most portable BSD that you can use on virtually all kinds of hardware.

	Kernel type	Installer	Default GUI	Package management	Config tools
FreeBSD	Monolithic with modules	Text	None	Ports	None
OpenBSD	Monolithic	Text	FVWM	Ports	Text
NetBSD	Monolithic with modules	Text	None	Pkgsrc	None
GhostBSD	Monolithic with modules	Graphical	Mate	Ports	None
DragonFly BSD	Hybrid	Text	None	DPorts	None
PC-BSD	Monolithic with modules	Graphical	KDE	PBI	Graphical

SUBSCRIBE

shop.linuxvoice.com

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

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

UK subs prices

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

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

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

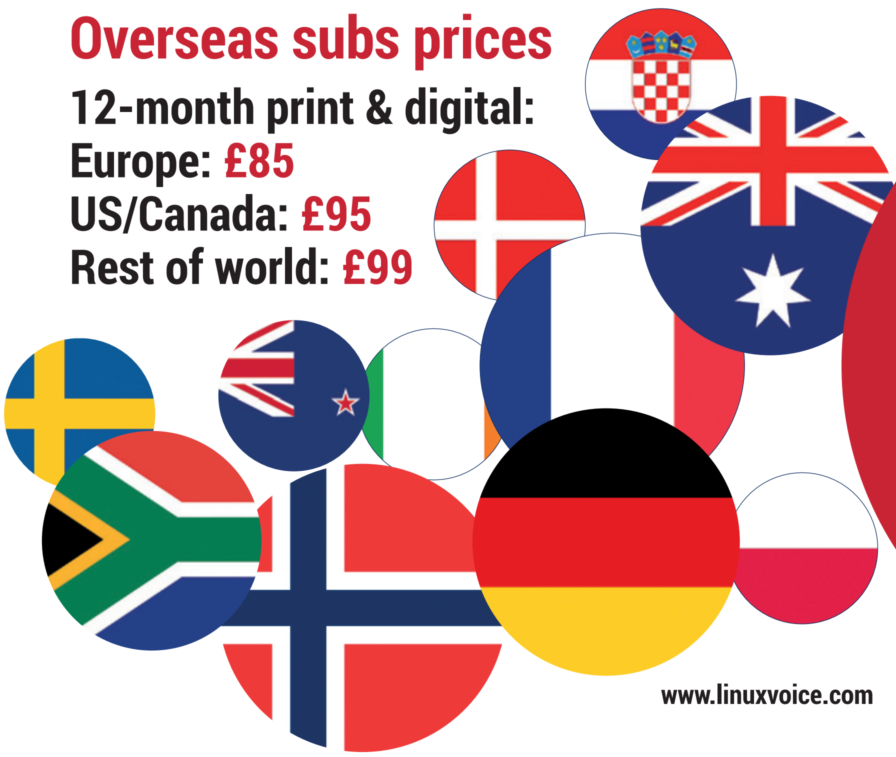
Save money on the shop price and get each issue delivered to your door

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

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

Overseas subs prices

12-month print & digital:
Europe: £85
US/Canada: £95
Rest of world: £99



DIGITAL SUBSCRIPTION*

ONLY £38

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

SYSADMIN: TERMINAL MULTIPLEXERS

We take a brief break from our normal our sysadmin coverage to reveal two of the most important commands you should learn: screen and tmux.

Many of us use the command line for lots of tasks – from the mundane to the complicated. But there's one single type of command that, if you're not already using it, will transform every command you type. This is a class of command known as the humble terminal multiplexer. You might be thinking that anything called a multiplexer might not sound so humble, but bear with us. It's not difficult to understand and it can make a huge difference to the way you use any Linux machine. The term 'multiplexer' comes from electronics jargon, where it's used to describe a device that transforms multiple inputs and a single output before being 'demultiplexed' into its composite parts at a destination.

With a terminal multiplexor on the command line, the multiple inputs are shell sessions and the single output is your display or single physical terminal. When described like this, it might not sound like the most exciting kind of command, but after you've got used to working within its environment, it's difficult to go back to a terminal that doesn't offer the same features; among a huge list, our favourites are the ability to spawn other terminals dynamically, switch between terminal sessions at any point, split the screen into different views and restore a running session at any point, even after disconnecting from the session completely. If you like running command line applications like *Mutt* for email, or *Irssi* for IRC, they're also perfect for persistent remote sessions.

They work best when you don't have access to a windowed environment, because it's easier to open multiple shell windows, although we'd argue less productive. The natural environment for a terminal multiplexer is the headless server. These are running somewhere remotely, and usually only accessible via SSH. You're often connected for long periods and often want to do more than one task at the same time. NAS boxes, Raspberry Pis, set-top-boxes and family PCs are equally well suited targets. A terminal multiplexer will allow you to open a single connection to any of these devices and use the same terminal session for multiple tasks. They should also allow you to disconnect from a session, complete with running processes, and re-connect at a later time.

They're synonymous with window managers in X – the part of your desktop that remembers where *Firefox* is running and lets you move the window around, restore its position when you reconnect or switch between running applications with a keyboard combination. They could even be thought of as the ultimate tiling window managers, albeit without visible tiles or window management, although with a little configuration, you can get close.

Screen test

This being open source there's more than one solution, and the two most popular are probably equally used and mostly matched for functionality. The first is the venerable *GNU Screen* and this is the command you're most likely to find pre-installed on your remote server or local NAS. Just type **screen -h** to check. If it's not there, *screen* will always be part of your distribution's principle repository or software archive. The other popular alternative is called *tmux*, a more descriptive name – terminal multiplexer. We're going to cover the basics of both so that you get a good idea of what they're capable of, and perhaps, which one you prefer. They start off working in much the same way before diverging slightly.

If we had to recommend just one terminal multiplexer, it would be *tmux*. It's a project that's still actively developed and had a few

Screen and tmux shortcuts

Function	screen	tmux
New Window	C	C
Next Window	N	N
Previous Window	P	P
Jump to window	0-9	0–9
Detach	D	D
Show shortcuts	?	?
List windows	"	S
Split vertically	Shift+S	"
Next region	Tab	cursor keys
Close region	Shift+X	X

more advanced features. But we're also covering *screen* because you might already have it installed.

Learning something of both also means that if you already know how to use one you should be able to migrate to the other tool by focusing on the similarities. This is particularly useful if you've always used *screen* and yet want to move to *tmux*. *Tmux* is often considered the more modern option, thanks to its better code and config files, but the ubiquity of *screen* means that this is the one you're more likely to have used already, and for that reason, this is the one most people already know about, but there's also lots both tools have in common, and looking at this commonality is the best place to start if you want to learn how to use either.

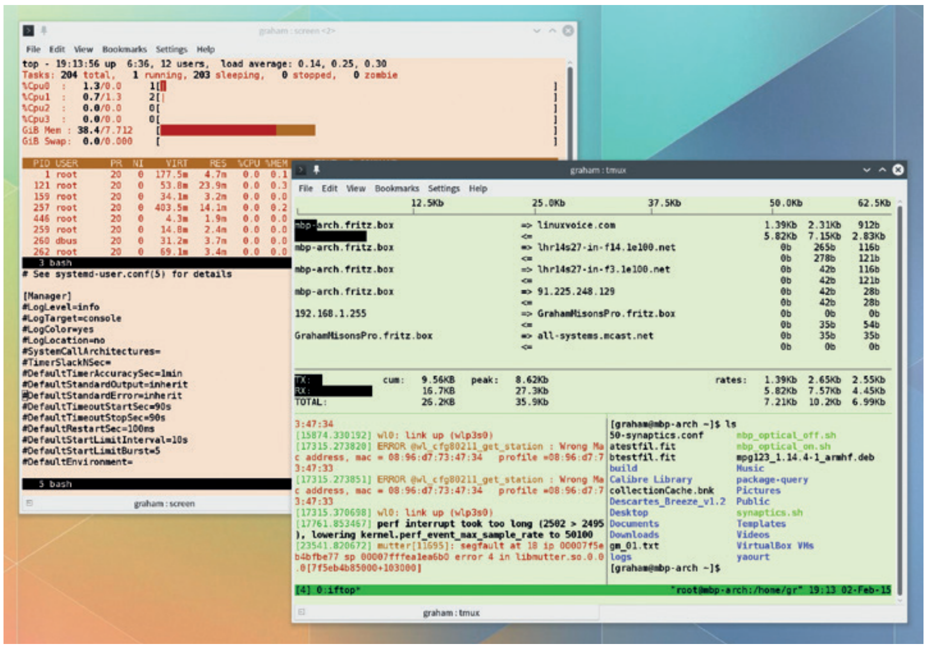
Both *screen* and *tmux* are launched by invoking their names on the command line. Screen will display a text-based splash screen asking you to press Space. After you've pressed Space and you're dropped back to the normal command line, it gives no other indication that *screen* is running in the background, other than changing the name of the session, usually visible in the window border at the top. And unless you use a special keyboard shortcut to summon *screen*'s control mode, the new session will perform exactly like a native session, and it's sometimes easy to forget whether you're operating within the confines of *screen*, or you've not run *screen* yet.

The tmux option

Tmux, by comparison, makes it much more obvious, because it places a coloured bar at the bottom of the session. Like the launch bar of a desktop, this can be configured to your preference, as much of the experience can be after you've become acquainted with the way that *tmux* works.

The super special secret escape sequence keyboard shortcut that puts both *screen* and *tmux* into their meta-command mode is slightly different: for *screen*, it's pressing Ctrl and A together. For *tmux*, it's Ctrl and B. What isn't immediately obvious, and something that will catch you out the first time you use either tool, is that there's no feedback to indicate the change of input mode that comes when you successfully hold down the keyboard shortcut. You just have to take it on faith that any further keys you press will be interpreted by either *screen* or *tmux*, rather than by the command line session you're running.

To give you some idea of what these tools are capable of, we'll run through how you might typically use either *screen* or *tmux*. From this silent entry point, press C to create a new session. If there was anything visible in your old session, this will disappear and new blank session will appear. But worry not! That previous session is still running and is still easily accessible. You may even have noticed that the status line at the bottom of *tmux* will have updated to show the presence of the new *Bash* terminal.



```
top - 19:13:56 up  6:36, 12 users,  load average: 0.14, 0.25, 0.30
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
%Cpu0  :   1.3/0.0     1[|                                          ]
%Cpu1  :   0.7/1.3     2[|                                          ]
%Cpu2  :   0.0/0.0     0[                                           ]
%Cpu3  :   0.0/0.0     0[                                           ]
GiB Mem : 38.4/7.712    [                                           ]
GiB Swap:  0.0/0.000    [                                           ]

  PID USER      PR  NI    VIRT    RES  %CPU %MEM
    1 root      20   0  177.5m   4.7m   0.0  0.1
  121 root      20   0   53.8m  23.9m   0.0  0.3
  159 root      20   0   34.1m   3.2m   0.0  0.0
  257 root      20   0  403.5m  14.1m   0.0  0.2
  446 root      20   0    4.3m   1.9m   0.0  0.0
  259 root      20   0   14.8m   2.4m   0.0  0.0
  260 dbus      20   0   31.2m   3.7m   0.0  0.0
  262 root      20   0   69.1m   3.4m   0.0  0.0
   3 bash
# See systemd-user.conf(5) for details

[Manager]
#LogLevel=info
#LogTarget=console
#LogColor=yes
#LogLocation=no
#SystemCallArchitectures=
#TimerSlackNSec=
#DefaultTimerAccuracySec=1min
#DefaultStandardOutput=inherit
#DefaultStandardError=inherit
#DefaultTimeoutStartSec=90s
#DefaultTimeoutStopSec=90s
#DefaultRestartSec=100ms
#DefaultStartLimitInterval=10s
#DefaultStartLimitBurst=5
#DefaultEnvironment=

   5 bash
```

```
             12.5Kb           25.0Kb           37.5Kb           50.0Kb      62.5Kb
mbp-arch.fritz.box               => linuxvoice.com                 1.39Kb  2.31Kb   912b
[redacted]                       <=                                5.82Kb  7.15Kb  2.83Kb
mbp-arch.fritz.box               => lhr14s27-in-f14.1e100.net          0b    265b    116b
                                 <=                                    0b    278b    121b
mbp-arch.fritz.box               => lhr14s27-in-f3.1e100.net           0b     42b    116b
                                 <=                                    0b     42b    121b
mbp-arch.fritz.box               => 91.225.248.129                     0b     42b     28b
                                 <=                                    0b     42b     28b
192.168.1.255                    => GrahamMisonsPro.fritz.box          0b      0b      0b
                                 <=                                    0b     35b     54b
GrahamMisonsPro.fritz.box        => all-systems.mcast.net              0b     35b     35b
                                 <=                                    0b      0b      0b

TX:  [redacted]   cum:   9.56KB   peak:   8.62Kb          rates:   1.39Kb  2.65Kb  2.55Kb
RX:  [redacted]         16.7KB           27.3Kb                    5.82Kb  7.57Kb  4.45Kb
TOTAL:                  26.2KB           35.9Kb                    7.21Kb  10.2Kb  6.99Kb
```

```
3:47:34                                            [graham@mbp-arch ~]$ ls
[15874.330192] wl0: link up (wlp3s0)               50-synaptics.conf      mbp_optical_off.sh
[17315.273820] ERROR @wl_cfg80211_get_station : Wrong Ma atestfil.fit     mbp_optical_on.sh
c address, mac = 08:96:d7:73:47:34   profile =08:96:d7:7 btestfil.fit     mpg123_1.14.4-1_armhf.deb
3:47:33                                            build                  Music
[17315.273851] ERROR @wl_cfg80211_get_station : Wrong Ma Calibre Library  package-query
c address, mac = 08:96:d7:73:47:34   profile =08:96:d7:7 collectionCache.bnk Pictures
3:47:33                                            Descartes_Breeze_v1.2  Public
[17315.370698] wl0: link up (wlp3s0)               Desktop                synaptics.sh
[17761.853467] perf interrupt took too long (2502 > 2495 Documents        Templates
), lowering kernel.perf_event_max_sample_rate to 50100 Downloads          Videos
[23541.820672] mutter[11695]: segfault at 18 ip 00007f5e gm_01.txt        VirtualBox VMs
b4bfbe77 sp 00007fffea1ea6b0 error 4 in libmutter.so.0.0 logs             yaourt
.0[7f5eb4b85000+103000]                            [graham@mbp-arch ~]$

[4] 0:iftop*                                "root@mbp-arch:/home/gr" 19:13 02-Feb-15
```

Both *tmux* and *screen* are similar, but *tmux* wins for splitting and user configuration.

Enter the escape sequence combo and press N. This is the terminal equivalent to switching to the next virtual desktop, and as we've only created two, this command will take you back to the previous session. You can switch between sessions directly by entering a number instead – sessions start at zero, and pressing P will take you back to your previous session. Now start something running, such as **top** and enter the escape sequence followed by D. This will 'detach' the terminal from all the sessions running within *screen* and *tmux*. You can now disconnect from the server, close the terminal, or go and make a cup of tea. Everything you started in your various sessions will keep executing.

There's some variation in the way that *screen* and *tmux* re-attach to a running session. Typing **screen -r** will re-attach to the running *screen* session, while *tmux a* will re-attach to *tmux*. If more than one session is running, *screen* will list them while *tmux* will simply connect to the first. If you want to specify a session, add the number value from the list. You can list running sessions in *tmux* by typing **tmux ls** and connecting to your chosen session with **tmux a -t session-name**. And that's all there is to the basics of using both tools. You can now start sessions and do all your usual command-line magic, safe in the knowledge you can disconnect and resume, and switch between any number of persistent terminal sessions. It's a serious upgrade to the way the terminal normally behaves.

"You can use tmux like a tiling window manager for the terminal"

Taking it further

Both *screen* and *tmux* will allow you to split the display, for example, so you can see two or more sessions at the same time. Use S in *screen* and double quotes in *tmux* for this. *Screen* won't automatically create a new terminal though, so you need to press the escape sequence followed by Tab to switch focus to the new region and then the escape sequence again, followed by C to create a new terminal.

This is where *tmux* starts to pull ahead, because it's more adaptable to both horizontal and vertical splits and how you can arrange them within sub-groups and move between them. You really can use *tmux* like a tiling window manager for the terminal, and it can be brilliant if you're reconnecting to view a server's logs, or building and developing code, or even if you're just copy files somewhere.

FOSSpicks

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

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

IRC Client

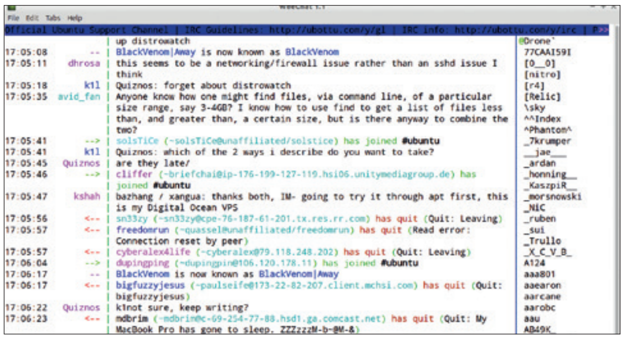
Weechat

IRC is so old it predates the web, yet it's still extremely popular because it works well, and it's widely supported by open source software. This means that anyone can quickly set up a server and have their own group chat. Alternatively, there are many public servers that you can create or join channels on. Almost every free software project has an IRC channel for developers and users to discuss the software. It's also the method of choice we use at Linux Voice for keeping in contact with each other while working on the magazine.

Weechat is a terminal-based client for IRC. In this age of graphical applications, a terminal-based client might sound old-fashioned, but actually, since IRC was designed as a text-based protocol in a world where graphical tools were a rare novelty, the protocol works well in a terminal, and many of the most widely used IRC clients are terminal-based.

IRC servers don't save histories of chats. That means that if you're client is not logged in to a server, then you can't go back and see what was said. This isn't always ideal. One solution many people use is to run a terminal-based IRC client on a server, then SSH into the server to access IRC. This works provided you have an SSH client on every machine you want to access IRC on. This is certainly possible given that even mobile phones can use SSH, but it's not always the most convenient option on mobile devices. *Weechat* can also act as a relay enabling you to connect another client to your *Weechat* session. For example, **www.glowing-bear.org** is a website that can connect to a *Weechat* client and give you access to your *Weechat* session from any machine with a web browser.

"Almost every free software project has an IRC channel for developers and users to discuss the software."



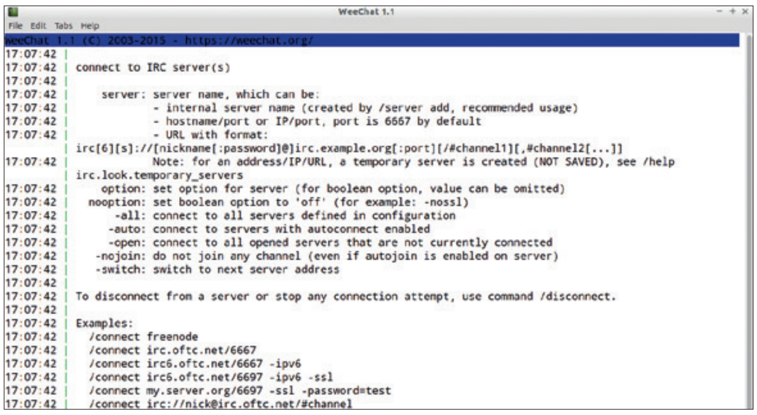
```
Official Ubuntu Support Channel | IRC Guidelines: http://ubottu.com/y/gl | IRC info: http://ubottu.com/y/irc | P>>
                  | up distrowatch                                                          @Drone`
17:05:08       -- | BlackVenom|Away is now known as BlackVenom                              77CAAI59I
17:05:11   dhrosa | this seems to be a networking/firewall issue rather than an sshd issue I [O__O]
                  | think                                                                   [nitro]
17:05:18      k1l | Quiznos: forget about distrowatch                                       [r4]
17:05:35 avid_fan | Anyone know how one might find files, via command line, of a particular  [Relic]
                  | size range, say 3-4GB? I know how to use find to get a list of files less \sky
                  | than, and greater than, a certain size, but is there anyway to combine the ^^Index
                  | two?                                                                    ^Phantom^
17:05:41      --> | solsTiCe (~solsTiCe@unaffiliated/solstice) has joined #ubuntu           _7krumper
17:05:41      k1l | Quiznos: which of the 2 ways i describe do you want to take?            __jae___
17:05:45  Quiznos | are they late/                                                          _ardan
17:05:46      --> | cliffer (~briefchai@ip-176-199-127-119.hsi06.unitymediagroup.de) has    _honning__
                  | joined #ubuntu                                                          _KaszpiR__
17:05:47    kshah | bazhang / xangua: thanks both, IM- going to try it through apt first, this _morsnowski
                  | is my Digital Ocean VPS                                                 _NiC
17:05:56      <-- | sn33zy (~sn33zy@cpe-76-187-61-201.tx.res.rr.com) has quit (Quit: Leaving) _ruben
17:05:57      <-- | freedomrun (~quassel@unaffiliated/freedomrun) has quit (Read error:     _sui
                  | Connection reset by peer)                                               _Trullo
17:05:57      <-- | cyberalex4life (~cyberalex@79.118.248.202) has quit (Quit: Leaving)    _X_C_V_B_
17:06:04      --> | dupingping (~dupingpin@106.120.178.11) has joined #ubuntu               A124
17:06:17       -- | BlackVenom is now known as BlackVenom|Away                              aaa801
17:06:17      <-- | bigfuzzyjesus (~paulseife@173-22-82-207.client.mchsi.com) has quit (Quit: aaearon
                  | bigfuzzyjesus)                                                          aarcane
17:06:22  Quiznos | k1not sure, keep writing?                                               aarobc
17:06:23      <-- | mdbrim (~mdbrim@c-69-254-77-88.hsd1.ga.comcast.net) has quit (Quit: My  aau
                  | MacBook Pro has gone to sleep. ZZZzzzM-b-@M-&)                          AB49K_
```

Freenode is one of the most popular IRC servers, and most major open source projects have a channel, including us (#linuxvoice).

There's comprehensive documentation in English, French, Japanese, German, Italian and Polish, so you should be able to get started without any problems. You'll find them at **https://weechat.org/doc/stable**.

One feature of IRC that's missing from most other chat protocols is the support for scripting, and to this end *Weechat* supports six scripting languages, so most programmers should find at least one they're familiar with. There's a repository of scripts at **https://weechat.org/scripts** that you can use if you don't want to write your own.

Although we have no figures to support this, we get the impression that *Weechat* is the fastest growing IRC client in terms of users. In the last couple of years, it's become widely talked about, and rightly so. It's an excellent client.



```
WeeChat 1.1 (C) 2003-2015 - https://weechat.org/
17:07:42 |
17:07:42 | connect to IRC server(s)
17:07:42 |
17:07:42 |     server: server name, which can be:
17:07:42 |             - internal server name (created by /server add, recommended usage)
17:07:42 |             - hostname/port or IP/port, port is 6667 by default
17:07:42 |             - URL with format:
         | irc[6][s]://[nickname[:password]@]irc.example.org[:port][/#channel1][,#channel2[...]]
17:07:42 |             Note: for an address/IP/URL, a temporary server is created (NOT SAVED), see /help
         | irc.look.temporary_servers
17:07:42 |     option: set option for server (for boolean option, value can be omitted)
17:07:42 |   nooption: set boolean option to 'off' (for example: -nossl)
17:07:42 |       -all: connect to all servers defined in configuration
17:07:42 |      -auto: connect to servers with autoconnect enabled
17:07:42 |      -open: connect to all opened servers that are not currently connected
17:07:42 |    -nojoin: do not join any channel (even if autojoin is enabled on server)
17:07:42 |    -switch: switch to next server address
17:07:42 |
17:07:42 | To disconnect from a server or stop any connection attempt, use command /disconnect.
17:07:42 |
17:07:42 | Examples:
17:07:42 |   /connect freenode
17:07:42 |   /connect irc.oftc.net/6667
17:07:42 |   /connect irc6.oftc.net/6667 -ipv6
17:07:42 |   /connect irc6.oftc.net/6697 -ipv6 -ssl
17:07:42 |   /connect my.server.org/6697 -ssl -password=test
17:07:42 |   /connect irc://nick@irc.oftc.net/#channel
```

As well as the various user guides, there's also built-in help that can be accessed with **/help <command>**.

PROJECT WEBSITE
https://weechat.org/

Pretend computer manipulator

Virtual Machine Manager

There's a wide variety of virtualisation software available. *VirtualBox* is great because the graphical interface makes it easy to learn to use. *Qemu* and *KVM* are really powerful, but it can take time to learn to use the command line tools. Linux containers (LXC) aren't true virtualisation, but can also be really useful if you know how to use them. Wouldn't it be great if there were a tool that brought *VirtualBox*'s ease of use to the power of *Qemu*, *KVM*, LXC and others? Well there is, and it's called *virt-manager* (or *Virtual Machine Manager*).

Red Hat developed *virt-manager* to help users easily take advantage of the full range of virtualisation options available on their enterprise distro, but it's open source and runs on most other distros as well. However, installing *virt-manager* can be a bit tricky because you have to make sure all the requisite components are installed, and that you have the right user permissions. You should look for distro-specific information before getting started.

Once up and running, *virt-manager* enables you to create new machines, edit their hardware, start and stop them, and all the other features you'd expect from a virtual machine manager. You can also use the specific options for the different types of virtual machine. For example, if you use *Qemu*, you can select from any of the architectures your install supports.

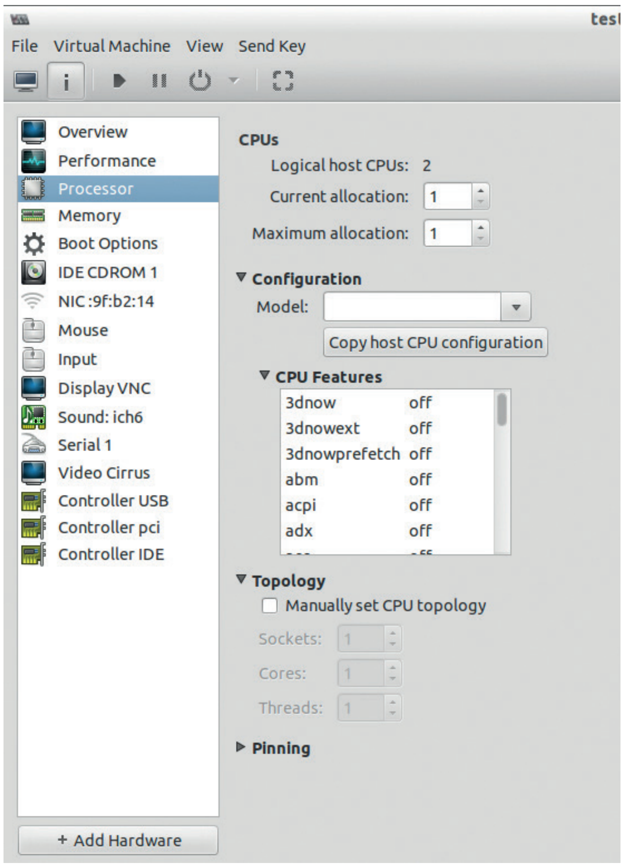
Once your machines are running, you can connect a display and use their graphical output, or connect to their serial console to see what's going on.

Perhaps the biggest disappointment with *virt-manager* is the documentation. Its website only contains a single piece, the FAQ, and this just contains three trivial questions. There's a bit more information on the man page, but this doesn't help much as it's a GUI application. Frankly, we expect a little better from Red Hat. Still, it's fairly straightforward to use, and most people should be able to get started without any problems.

Management layer

Virt-manager is in reality just a graphical layer that sits on top of *libvirt*. It's this library that manages the communication between all the different hypervisors. Since it doesn't actually run the virtual machines (it's just used to manage them), it can distance itself from the virtual machines to a certain degree. For example, you can use *virt-manager* to start a virtual machine, then close *virt-manager* and the virtual machine will continue running. You can then reopen *virt-manager* and use it to shut down the VM. This architecture also means that you can connect it to virtual machines running on different physical computers, provided you have the appropriate permissions. The machines don't have to have been created by virt-manager: as long as they're running on one of the supported platforms, then you can connect to them.

While *virt-manager* is great for setting up machines and getting them running, it's not so great for managing whole groups of them. There are whole other toolsets designed for just this task (such as the ones that come with CoreOS). But for simple use, especially desktop use, *virt-manager* is an excellent tool for taking the pain out of virtual machines.

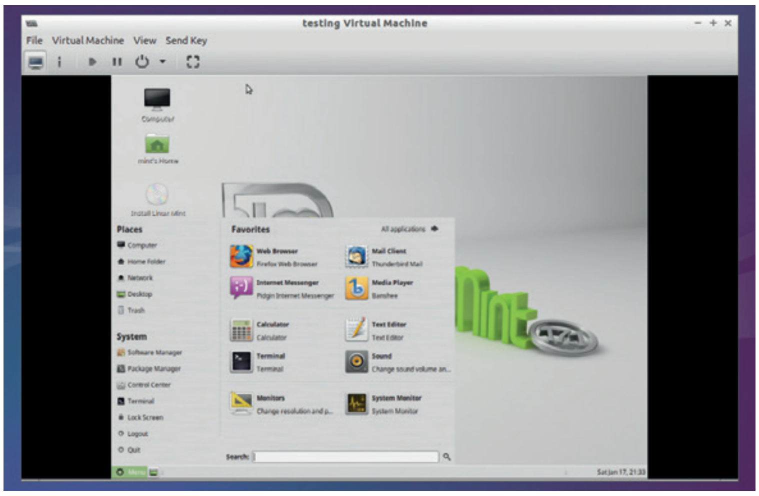


CPUs
Logical host CPUs: 2
Current allocation: 1
Maximum allocation: 1
Configuration
Model:
CPU Features

3dnow	off
3dnowext	off
3dnowprefetch	off
abm	off
acpi	off
adx	off

Topology
Manually set CPU topology
Sockets: 1
Cores: 1
Threads: 1

The huge range of configuration options in *virt-manager* allows you to tailor almost every aspect of your virtual machines.



Virtualisation allows you to try out many distros without rebooting your machine.

PROJECT WEBSITE
https://virt-manager.org/

Python 3D library

Soya 3D

Most modern computers – and even most phones – come with some form of 3D graphical acceleration. This hardware enables the computer to render incredibly complex scenes in real time. OpenGL is the standard rendering library for Linux, and it's very powerful, but it's also quite low-level and requires the programmer to use vector-matrix algebra to manipulate the scene.

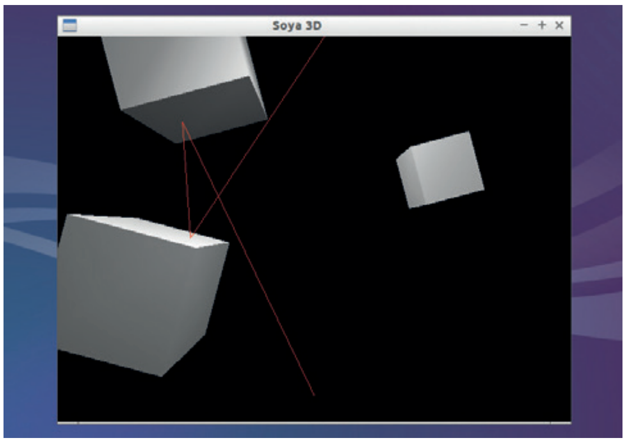
Soya 3D is a Python library that aims to free the programmer from this low-level detail while still getting the advantage of accelerated graphics. It lets you import models created in 3D modelling software (such as *Blender*) and manipulate them in your Python code.

Soya's stand-out feature is the number of really well commented tutorials that make it easy for anyone who knows Python to get started with 3D programming. You can get them from **http://home.gna.org/oomadness/en/soya3d/tutorials**. These take you through everything from creating a simple scene with just a single object in it, to creating complex animated scenes with interacting objects and varied lighting effects.

Accessible modelling

However, 'tutorials' is a bit of a misnomer. In reality, they're very well commented examples, but someone with basic knowledge of Python should find it quite easy to follow them and understand how *Soya* works. The only caveat with the tutorials is that they haven't been updated in quite some time, so they don't include all the latest features of *Soya 3*. However, they should include enough information to get you started and help you understand the principals behind the library.

If you want to get into 3D programming, *Soya* is a great place to start, because while it abstracts out a lot of the detail, it follows the same process as lower-level toolkits, so if you want to move on, you should find it quite easy.



Impress friends and colleagues with complex-looking 3D graphics – just don't tell them how easy they are to code.

PROJECT WEBSITE
www.lesfleursdunormal.fr/static/informatique/soya3d/index_en.html

Disk-saving GUI

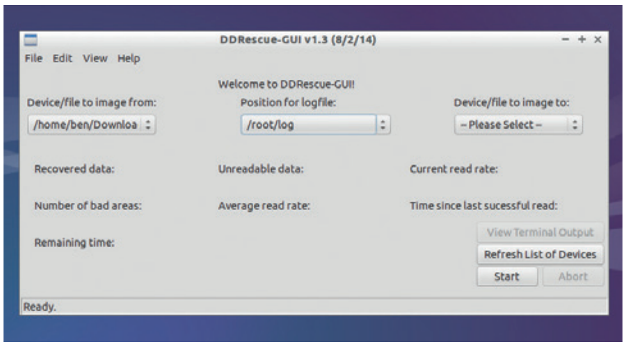
DDRescue-GUI

One of our most-used command line tools when it comes to manipulating disks is **dd**. This is a great tool for copying data to and from block devices such as hard drives and USB sticks. It's an old Unix tool, and the origins of its name have been lost in the mists of time (though some people claim it means 'data destroyer' because a misplaced argument can wipe all the data off a disk. This tendency to wipe data, provided the impetus for **ddrescue**, a newer program with more options to make it better suited to fetching data off a block device that's in the process of breaking. It enables you to specify the read order, and try particular blocks multiple times until you get a successful read. It's arguably the best tool for recovering data from a failing drive.

Unless you're very unlucky, you won't experience failing drives very often, and that means that you won't need **ddrescue** very often. The problem with command line tools that you don't use very often is that it can be hard to remember the correct options. In a situation where you've got a failing drive, missing an option can mean data is lost forever. *DDRescue-GUI* is a simple GUI wrapper for **ddrescue** that enables you to view and select the appropriate options, thus eliminating the risk of forgotten options.

As with all rescue tools, the worst time to learn it is when you need to use it, so it's worth trying out this software to make sure you're familiar with it should the worst happen. When a drive fails, it can deteriorate rapidly, and the longer you spend working out how to use the software, the more data you could lose. Once you've got the data off the broken disk, you can then spend as long as you need figuring out how to extract the information you need from the image file.

"The worst time to learn a rescue tool is when you need to use it."



Welcome to DDRescue-GUI!
Device/file to image from: /home/ben/Downloa
Position for logfile: /root/log
Device/file to image to: – Please Select –
Recovered data:
Unreadable data:
Current read rate:
Number of bad areas:
Average read rate:
Time since last sucessful read:
Remaining time:
Ready.

As well as recovering data, you could use *DDRescue-GUI* as a graphical tool for writing an ISO image to a USB stick to make a live distro, or for backing up drives.

PROJECT WEBSITE
https://launchpad.net/ddrescue-gui

JavaScript framework

IO.js

IO.js describes itself as a spork of Node.js, but that just adds more confusion, not less. A spork – for those of you unfamiliar with the term – is like a fork, but it's designed to supersede the original project so the two end up merging. This is in contrast to a true fork, which is designed to take the codebase in a new direction.

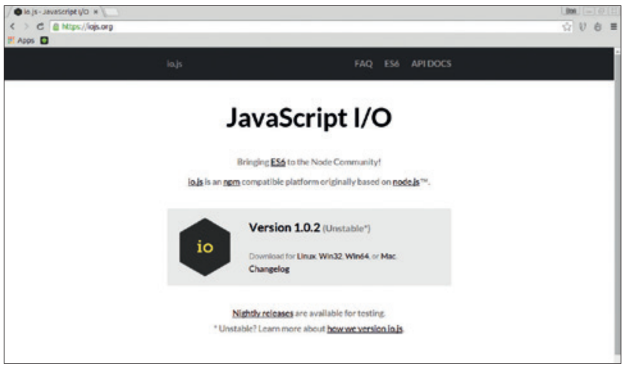
So that's a spork, but what's Node.js, and why would anyone want to spork it? Node.js is a JavaScript runtime that's designed for running event-driven, non-blocking applications. It's become particularly popular for building the back-ends for interactive web apps.

Node.js is open source, however, under the aegis of its current steward, an organisation called Joyent, development of Node.js appears to have stalled. Although there still appear to be people working on it, the pace of releases has slowed down.

For quite some time, Node.js has been based on an unsupported version of Google's V8 JavaScript engine. The people behind IO.js hope to increase the pace of releases, and they're already working on a more up-to-date version of V8 including support for the latest features in EMCA Script 6 (ES6). Unlike Node.js, IO.js isn't run by a company, but by developers from the community. The hope is that this will mean the project will not be dominated by a single party.

IO.js is compatible with the Node Package Manager, so many projects should work on both Node.js and IO.js at the moment, although this may change if the two projects diverge further. It may be slightly confusing to some new users that Node.js (the older project) is on version 0.10, while IO.js has just released version 1.0. This isn't intended to signify that IO.js is more mature; they're just using a different release numbering system and using 1.0 to show that they're a fully independent project.

"The people behind IO.js hope to increase the pace of releases."



JavaScript I/O

Bringing ES6 to the Node Community!

io.js is an npm compatible platform originally based on node.js™.

Version 1.0.2 (Unstable*)

Download for Linux, Win32, Win64, or Mac

Changelog

Nightly releases are available for testing.

* Unstable? Learn more about how we version io.js.

Forks, spoons and sporks? Cutlery gone mad, or the spirit of democracy in open source software?

PROJECT WEBSITE
https://iojs.org

Music manager

Quod Libet

There are some really good music players for Linux, so in order to stand out a player has to do something special. In *Quod Libet*'s case, that something special is its search feature. It allows you to search tags, and also combine different tags, negate particular search options and others. Basically, it enables you to very easily specify exactly what music you want on your playlist.

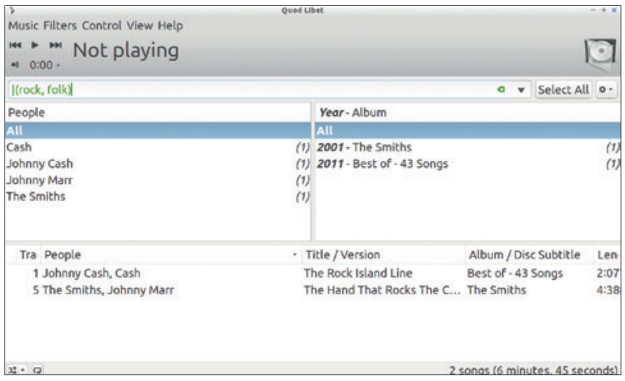
For example, do you want all rock music, but want to keep it short and snappy? Well, you can search for all songs that have the tag rock, but are less that five minutes long. How about all pop music released before 1979? No problem. This is all done through *Quod Libet*'s text-based search language, detailed at **https://quodlibet.readthedocs.org/en/latest/guide/searching.html**

Quod Libet works on the principal that you know wha you want to listen to, so gives you the power to specify it. This is in direct contrast to some other music players that try to guess what you want and make the decision for you.

Of course, all this requires you to have a good source of properly tagged music, otherwise *Quod Libet* will struggle to find the songs you specify. *Quod Libet* does come with some tools to help you tag you files, but it doesn't support automatic tagging. Provided all your songs are well tagged, *Quod Libet* will easily scale to collections with large numbers of songs. In fact, its powerful search features are probably best suited to albums with thousands of songs or more.

The latest version (3.3) comes with a few new features including better handling of diacritic marks in searches, easier access to song lyrics (via **lyrics.wikia.com**) and support for more file formats.

If you've been struggling to manage a large audio collection on Linux, and you're comfortable using text-based search tools, *Quod Libet* could be the music player you've been waiting for.



|(rock, folk)

People	
All	
Cash	(1)
Johnny Cash	(1)
Johnny Marr	(1)
The Smiths	(1)

Year - Album	
All	
2001 - The Smiths	(1)
2011 - Best of - 43 Songs	(1)

Tra	People	Title / Version	Album / Disc Subtitle	Len
1	Johnny Cash, Cash	The Rock Island Line	Best of - 43 Songs	2:07
5	The Smiths, Johnny Marr	The Hand That Rocks The C...	The Smiths	4:38

2 songs (6 minutes, 45 seconds)

Powerful functionality, not graphical frivolity, is what makes *Quod Libet* stand out.

PROJECT WEBSITE
https://code.google.com/p/quodlibet

Image compressor

Mozjpeg

Images: without them, the web would be a boring place. Whether they're cute kittens, flashy logos, or textured backgrounds, images are what change web page from piles of text into an enjoyable medium. However, they take up bandwidth, and that means they slow down page loads, and cost more money.

You can – and should – compress images. However, the most common form of image compression is JPEG, and that's over 20 years old. There are better options, but the nasty world of patent enforcement means they're rarely used.

Mozjpeg isn't a new compression standard, but an optimised JPEG compression tool using newer techniques behind the scenes to squeeze images down to the smallest possible size while retaining as much detail as possible. The gains aren't huge (around 5–10%), but if images take up a significant portion of your web traffic, this could be a meaningful difference. After all, a 5% reduction in bandwidth bills and storage requirements could result in a huge saving for popular websites. Similarly, a 5% speedup in page load times will result in a better experience for your users.

The images from *Mozjpeg* are completely compatible with normal JPEG decoders, so you can view them in older web browsers without any plugins or additional codecs.

Mozilla, the creator of the tool, is keen to point out that *Mozjpeg* is specifically tuned for web images, and isn't intended as a general purpose JPEG tool. If you're planning on printing images, then you should stick with a standard JPEG library, as this will give better images at this quality.

"Mozjpeg is an optimised JPEG compression tool for the web."

One of these images is compressed 9% more than the other. Can you tell which one? We can't, but we know that one loads faster.

PROJECT WEBSITE
https://github.com/mozilla/mozjpeg

Graphical command tool

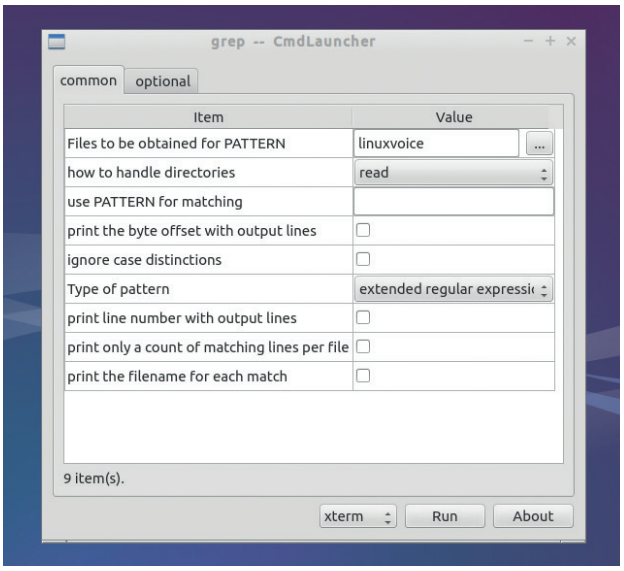
CmdLauncher

The Unix shell interface is an amazingly powerful thing. It's arguably the most powerful interface ever created for computing. However, it can be intimidating for new users. *CmdLauncher* is a GUI for command line tools to allow new users to use things like **grep** without delving down into the text interface.

CmdLauncher isn't built for a specific set of commands; instead it uses configuration files created for each command. These are fairly straightforward and can easily be created for new commands. It only comes with two: **grep** and **upx** (a tool for creating compressed executable files).

Another way of looking at *CmdLauncher* is as a GUI builder for text-based tools. If you need to help someone use some particular arcane command line tool, you can quickly create a config file that means they can use the tool in a friendly window without having to know what's going on underneath. By deciding on the particular options you make available, and how you name them, you can help them use particular tasks. You could even use it to create an interface for any shell scripts you create. If you do create useful config files, be sure to share them as it would be helpful if the software came with more than the two examples.

CmdLauncher can only be used for individual commands, and you can't set up chains of *CmdLauncher* instances that each pipe data into the next, so it's not a complete replacement for the shell interface. Advanced users will always have to run the command line, but this is just powerful enough to help some people who aren't interested in computing get things done without making the plunge to the shell.



Item	Value
Files to be obtained for PATTERN	linuxvoice
how to handle directories	read
use PATTERN for matching	
print the byte offset with output lines	☐
ignore case distinctions	☐
Type of pattern	extended regular expressi
print line number with output lines	☐
print only a count of matching lines per file	☐
print the filename for each match	☐

9 item(s).

Never before has it been so easy to get started with **grep**.

PROJECT WEBSITE
http://cmdlauncher.topbug.net

FOSSPICKS Brain Relaxers

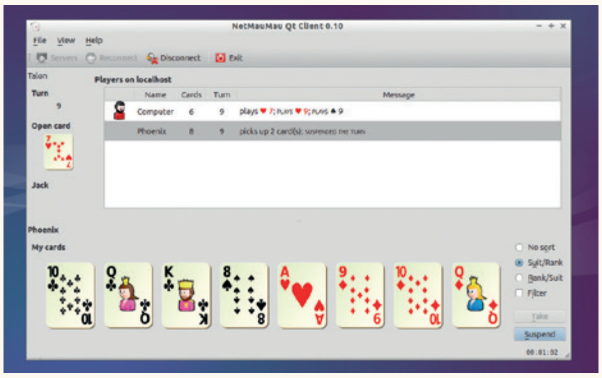
Online card game

NetMauMau

The card game Mau Mau is similar to the game Uno: you have to get rid of all your cards by playing a card of the same suit or number as the one previously played. A few cards have special rules, but there doesn't seem to be a general consensus on which cards have what special rules. Instead, it's common for different groups of people to play by slightly different rules. The makers of *NetMauMau* outline their rules on the GitHub page (**https://github.com/velnias75/NetMauMau**) and far be it from us to claim these are wrong. If you disagree, there are a few command line options to change the rules to something that suits you better.

NetMauMau is a client–server based game that you can play either with other players across the internet, or with a computer AI on the network, and there can be up to five people in each game.

We didn't find the AI especially challenging, but we should point out that the reviewer spent many nights perfecting his Mau Mau skills in Tanzania where he didn't have a TV or computer, so had nothing else to do apart from create increasingly complex techniques for shedding cards in Last Cardi (as Mau Mau is known in Swahili). The trick, as with so many games, is to always think not of what you can play now, but what you'll want to play in the future. There's more to this game than it initially seems. The one thing missing from *NetMauMau* is any way to search for public servers. If you fancy a game, you're limited to either playing against AI, or locating opponents for yourself. Perhaps this is for the best, as our memory of Mau Mau is as a social game, and that would be lost on people you didn't know.



Players on localhost

Name	Cards	Turn	Message
Computer	6	9	plays ♥ 7; PLAYS ♥ 9; PLAYS ♠ 9
Phoenix	8	9	picks up 2 card(s); SUSPENDED THE TURN

The internet has a full list of optional rules, though few of them are implemented in *NetMauMau*.

PROJECT WEBSITE
nhttps://github.com/velnias75/NetMauMau

Racing game

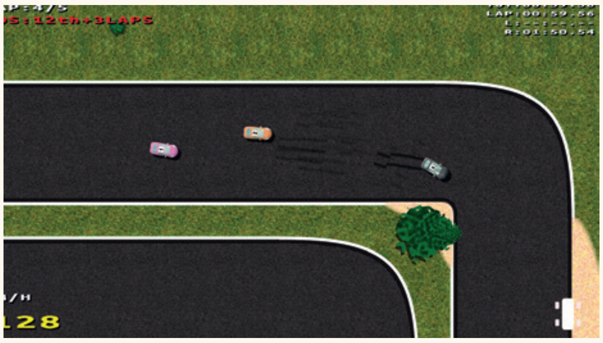
Dust Racing 2D

Car racing games are hard to do well. To feel right, they need to have smooth graphics, physics that work properly, and AI players that present the right levels of difficulty. *Dust Racing 2D* is a top-down car racing game that gets all these right.

It sometimes feels a little like you're driving on an oil slick, and learning to handle the car is a key part of advancing in the game. It claims to be a 2D game, but this isn't quite right. Although it doesn't have full 3D, it does have some effects that gives a 3D feel to the top-down view. Our test system has Intel HD graphics and no graphics card, and on this it ran smoothly and looked great, so you don't need a top-spec graphics card or proprietary blobs to play.

The controls are simple (by default just the arrow keys), so it's quick to get started, even for people not used to racing games. There are five tracks to play by default (although only one is unlocked until you prove your racing chops), or you can design your own with the level editor.

Overall, *Dust Racing 2D* isn't a complex game, but it is great fun to slide around corners and bounce off your competitors in a mad dash to the finish. There are various two-player options (regular race, duel and time trials), so you can challenge your mates to some open-source racing fun. LV



Car control isn't our strong point, but what we lack in finesse, we more than make up pure speed.

"It's great fun to slide round corners and bounce off your competitors."

PROJECT WEBSITE
http://sourceforge.net/projects/dustrac/

FORTHCOMING EVENTS

DATES FOR YOUR DIARY

DEVOPS Spring 2015 • 24th, 25th & 26th March 2015
The Hilton York, 1 Towers Street, York YO1 9WD

Tuesday 24th March - Workshops

'Zero to Perl' half-day - provided by Shadowcat Systems Ltd.
'Practical Digital Forensics' half-day - tutor: Tim Fletcher
...Further Workshops to be Announced...

Wednesday 25th & Thursday 26th - Conference
provisional list of accepted talks to date:

Julien Pivotto - ***Shipping your product with Puppet code***
Stephen Quinney - ***Intrusion Detection using the Linux Audit System***
Toshaan Bharvani - ***Virtual Machine Lifecycle Management with Anisible***
Kenneth MacDonald - ***Kerberos – Protocol and Practice***
Wim Godden - ***Intrusion detection through backups (and other securtiy tricks)***
Pieter Baele - ***Linux centralized identity and authentication interoperability with AD***
Stu Teasdale - ***Beyond Blue-Freen: Migrating a legacy application to CI and the Cloud***
2nd Quadrant - ***State of PostgreSQL Database 2015***
Bernd Erk - ***Open source Monitoring with Icinga***
Mark Cairney - ***Federated Access Management***
Peter Tribble - ***Creating Tribblix***
Richard Melville - ***An Introduction to Btrfs***
Nick Moriarty - ***Puppet as a Legacy system***
Mark Keating - ***Who's The Pan*** plus many more...

for more information see: http://www.flossuk.org/Events/Spring2015
Book your place now! Use code 'VOICE15' for a 15% discount on Conference fees.
This event is sponsored by: ELIGO Recruitment

OpenTech 2015 • Saturday 13th June 2015
ULU, University of London Union

OpenTech will be 10 years old in 2015.
This event is only possible because of wonderful and generous sponsors in the past.... If you or your company is interested in Sponsoring please get in touch - ***opentech@opentech.org.uk***

The event's predecessors were low cost, one-day conferences about technologies that anyone can have a go at, from 'Open Source' style ways of working to repurposing everyday electronics hardware.

http://www.opentech.org.uk

Dynamic Languages Conference - Saturday 20th June 2015 - Manchester
for more information see: http://www.dynamiclanguages.co.uk

If you are not a member and want to be kept informed of future events please subscribe to '***announce@ukuug.org***' see: ***http://lists.ukuug.org/mailman/listinfo/announce***

UKUUG - Bronze Sponsor members include: SUSE

Why not join today? Annual Individual membership is just £42.00 inc. VAT. See: ***http://www.flossuk.org/join***
As a member you can attend all our events at the specially discounted member rates and receive our quarterly Newsletter!

UKUUG Ltd. t/a FLOSS UK, The Manor House, Buntingford, Herts SG9 9AB
Tel: 01763 273475 Email: office@ukuug.org

Follow FLOSS UK on twitter or Facebook

LINUX VOICE

TUTORIALS

Dip your toe into a pool full of Linux knowledge with nine tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
is using technology as an excuse not to venture out into the cold, dark world.

This month I've been mostly playing with the new Raspberry Pi. It's quite an impressive piece of kit, and it's joined my collection of small ARM boards along with the Udoo, Odroid, Matrix TBS, BeagleBone Black, and others. In fact, I've owned more ARM computers than x86 – and that's excluding embedded devices, mobile phones and tablets.

This means most of my processors were designed right here on this precious stone set in the silver sea (as Shakespeare put it). However, the fact that they were designed in Britain isn't as important as the fact that they weren't designed in America.

Has Silicon Valley lost its touch? To take just one example, the most innovative new phone systems are no longer American (Sailfish from Finland, Tizen from Korea and Ubuntu Touch from the UK). OK, you could also include Firefox OS in that list, but even though Mozilla is based in America, it's international and the phones aren't targeted at the USA.

It's starting to feel like the only things left in the Valley are outdated but entrenched companies, and new startups creating overvalued websites. That doesn't matter though, because the rest of the world is more innovative than it's ever been.
ben@linuxvoice.com

In this issue...

LINUX VOICE TUTORIAL

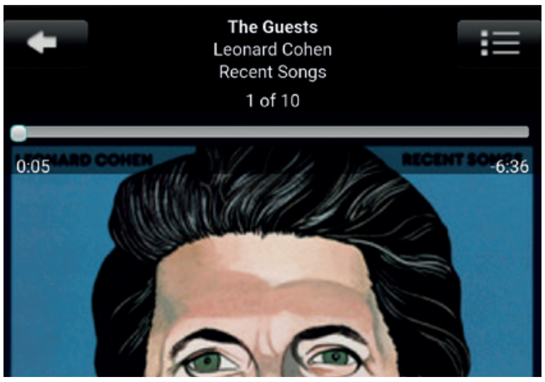
SIX STEPS TO AUDIO STREAMING PARADISE

GRAHAM MORRISON

Install and configure the best audio streaming solution we've found: Logitech Media Server.

WHY DO THIS?
- Stream music from one source throughout the whole house.
- Put dusty old hardware to work as a music client.
- Never be without your faourite tunes.

There used to be a product not so long ago that enabled you to simply run a custom developed and open source music server on a PC or NAS and plug the player into an amplifier, and get complete access to your music. The company was called Slim Devices and its product was called a Squeezebox. In 2006, the company was bought by Logitech, which continued to make lovely boxes until last year, when the whole project was canned. Fortunately, the server was open source and has continued to be developed. This is a wonderful thing, because it's the best piece of music streaming software we've found – and to demonstrate this, we're going to get it up and running in just six easy steps.



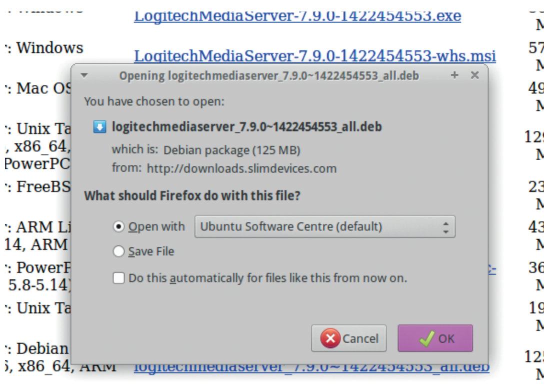
Logitech Media Server (*LMS*) is a brilliant solution for controlling and playing music from an Android app.

Step by step: Music streaming with Logitech Media Server

1 Install the server

The server component manages your local music collection, streams your music to clients, and offers a web interface and a way to remotely control your music playback. We're going to install the latest server package on Xubuntu 14.10, but you should find the process similar for many other distributions, especially as we're installing the package ourselves.

As Logitech no longer supports the product, its official repositories are now out of date, so we grabbed the Debian Installer Package version of the latest bleeding-edge community release (7.9.0) from **http://downloads.slimdevices.com/nightly**. This is a Deb file that can be installed on Ubuntu derivatives with a simple click – though you'll need to accept the caveat that you're installing a package from an insecure source. The command line equivalent is to type **sudo dpkg -i** followed by the package name.



LogitechMediaServer-7.9.0-1422454553.exe

: Windows LogitechMediaServer-7.9.0-1422454553-whs.msi

Opening logitechmediaserver_7.9.0~1422454553_all.deb

You have chosen to open:

logitechmediaserver_7.9.0~1422454553_all.deb

which is: Debian package (125 MB)
from: http://downloads.slimdevices.com

What should Firefox do with this file?

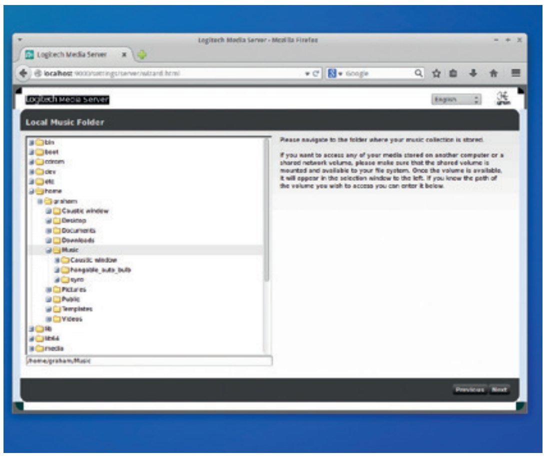
[✓ Open with] Ubuntu Software Centre (default)
Save File
Do this automatically for files like this from now on.

Cancel OK

logitechmediaserver_7.9.0~1422454553_all.deb

2 Setting up the server

If you point a local web browser at **localhost:9000** (or replace **localhost** with the IP address of your server for LAN access), you'll be presented with the first page of the startup wizard. The first page asks you to create an account with **mysqueezebox.com**, but we'd recommend skipping this step and pressing Next. The next page asks for the location of your music collection, so you need to navigate to the root folder of your files. You can also install plugins onto Media Server to give you access to music from internet radio, iPlayer, YouTube and subscription services like Spotify. The next step asks where you keep your playlists, and you can make this the same as your music folder. Click on Finish and the server will start scanning your collection and creating a database.



Logitech Media Server - Mozilla Firefox

localhost:9000/settings/server/wizard.html

Local Music Folder

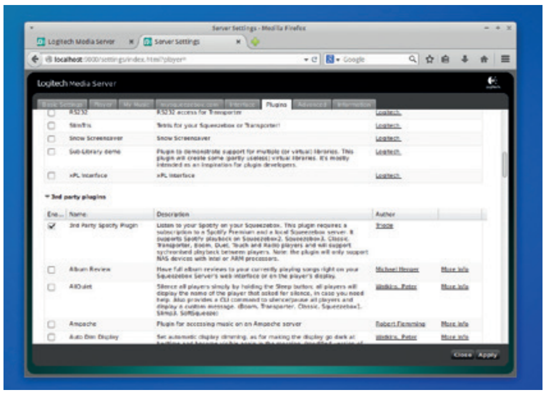
Please navigate to the folder where your music collection is stored.

If you want to access any of your media stored on another computer or a shared network volume, please make sure that the shared volume is mounted and available to your file system. Once the volume is available, it will appear in the selection window to the left. If you know the path of the volume you wish to access you can enter it below.

/home/graham/Music

3 Add some plugins

The server's web page should now transform itself into the playing interface with two panels. On the left panel you'll find your music sources, including your own collection within 'My Music'. to the right, you find the area that holds your dynamic playlist. It will show track names and images as you add and play them. If you want to add extra sources, click on the Settings button on the bottom-right. This is where you can change everything about your server installation, but for now, just click on Plugins. If you use Spotify, we'd highly recommend Triode's unofficial plugin. If you're in the UK, the iPlayer plugin is also excellent for catching up on BBC radio. Some plugins will need the server to be restarted, and if you need to add login credentials, such as for Spotify, you can do this by clicking on Customise in this list afterwards.



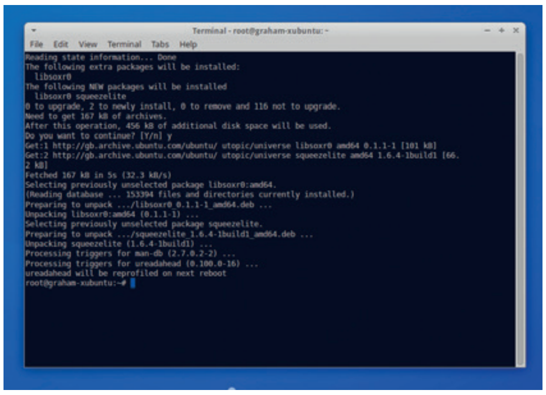
☐	RS232	RS232 access for Transporter	Logitech	
☐	SlimTris	Tetris for your Squeezebox or Transporter!	Logitech	
☐	Snow Screensaver	Snow Screensaver	Logitech	
☐	Sub-Library demo	Plugin to demonstrate support for multiple (or virtual) libraries. This plugin will create some (partly useless) virtual libraries. It's mostly intended as an inspiration for plugin developers.	Logitech	
☐	xPL Interface	xPL Interface	Logitech	

▾ 3rd party plugins

Ena...	Name	Description	Author	
✓	3rd Party Spotify Plugin	Listen to your Spotify on your Squeezebox. This plugin requires a subscription to a Spotify Premium and a local Squeezebox server. It supports Spotify playback on Squeezebox2, Squeezebox3, Classic, Transporter, Boom, Duet, Touch and Radio players and will support synchronised playback between players. Note: the plugin will only support NAS devices with Intel or ARM processors.	Triode	
☐	Album Review	Have full album reviews to your currently playing songs right on your Squeezebox Server's web interface or on the player's display.	Michael Herger	More info
☐	AllQuiet	Silence all players simply by holding the Sleep button; all players will display the name of the player that asked for silence, in case you need help. Also provides a CLI command to silence/pause all players and display a custom message. (Boom, Transporter, Classic, Squeezebox1, SB1/3, SoftSqueeze)	Watkins, Peter	More info
☐	Ampache	Plugin for accessing music on an Ampache server	Robert Flemming	More info
☐	Auto Dim Display	Set automatic display dimming, as for making the display go dark at	Watkins, Peter	More info

4 Install the client

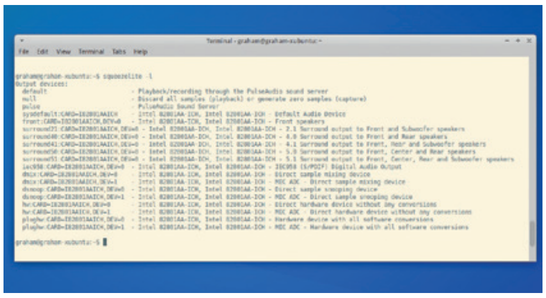
It's not time to connect your client hardware to your network and your speakers and it would discover the server and allow you access to your music through the bundled remote control. Fortunately, there are two open source projects that re-create the hardware in software, enabling you to add as many clients as you want. You can even install them into the same machine as the server. One option is called *SoftSqueeze*. This is a Java-based tool that recreates the look and feel of the hardware. We like it. But we prefer a more lightweight option called *Squeezelite*. This is a command-line tool that takes little resources. Most repositories will have *Squeezelite* available from their default repositories and you'll need to install this along with **flac**, **libfaad2** and **libmad0** packages for Flac, AAC and MP3 support.



```
Reading state information... Done
The following extra packages will be installed:
  libsoxr0
The following NEW packages will be installed
  libsoxr0 squeezelite
0 to upgrade, 2 to newly install, 0 to remove and 116 not to upgrade.
Need to get 167 kB of archives.
After this operation, 456 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://gb.archive.ubuntu.com/ubuntu/ utopic/universe libsoxr0 amd64 0.1.1-1 [101 kB]
Get:2 http://gb.archive.ubuntu.com/ubuntu/ utopic/universe squeezelite amd64 1.6.4-1build1 [66.
2 kB]
Fetched 167 kB in 5s (32.3 kB/s)
Selecting previously unselected package libsoxr0:amd64.
(Reading database ... 153394 files and directories currently installed.)
Preparing to unpack .../libsoxr0_0.1.1-1_amd64.deb ...
Unpacking libsoxr0:amd64 (0.1.1-1) ...
Selecting previously unselected package squeezelite.
Preparing to unpack .../squeezelite_1.6.4-1build1_amd64.deb ...
Unpacking squeezelite (1.6.4-1build1) ...
Processing triggers for man-db (2.7.0.2-2) ...
Processing triggers for ureadahead (0.100.0-16) ...
ureadahead will be reprofiled on next reboot
root@graham-xubuntu:~#
```

5 Run the client

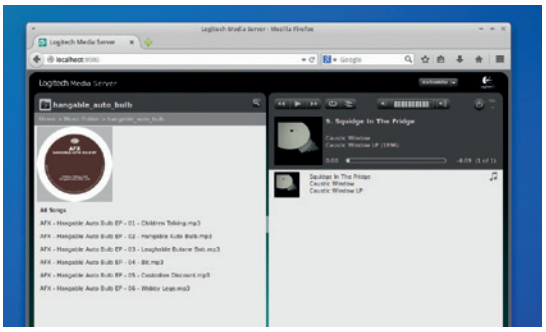
Our client of choice, *Squeezelite*, needs to be told what audio interface to use and the address of your server. You can list all detected sound devices by typing **squeezelite -l**. If your system is running *PulseAudio*, you should see this listed as 'pulse' and we'd recommend using it. The argument for telling *Squeezelite* which audio device to use is **-o "EXACT_DEVICE_NAME**'. While you can also tell *Squeezelite* the specifics of your audio interface's capabilities, it always will try the best option based on the source material. You'll need to add **-s IP_ADDRESS** to tell the client the IP address of your server – 'localhost' works too if you're running the client on the same machine as the server, and we like to provide a name for the player with the **-n** argument. Here's our invocation for reference: **squeezelite -o pulse -s localhost -n xubuntu** (adding **-z** will run the player in the background, silently as a daemon).



```
graham@graham-xubuntu:~$ squeezelite -l
Output devices:
  default                       - Playback/recording through the PulseAudio sound server
  null                          - Discard all samples (playback) or generate zero samples (capture)
  pulse                         - PulseAudio Sound Server
  sysdefault:CARD=I82801AAICH   - Intel 82801AA-ICH, Intel 82801AA-ICH - Default Audio Device
  front:CARD=I82801AAICH,DEV=0  - Intel 82801AA-ICH, Intel 82801AA-ICH - Front speakers
  surround21:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - 2.1 Surround output to Front and Subwoofer speakers
  surround40:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - 4.0 Surround output to Front and Rear speakers
  surround41:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - 4.1 Surround output to Front, Rear and Subwoofer speakers
  surround50:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - 5.0 Surround output to Front, Center and Rear speakers
  surround51:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - 5.1 Surround output to Front, Center, Rear and Subwoofer speakers
  iec958:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - IEC958 (S/PDIF) Digital Audio Output
  dmix:CARD=I82801AAICH,DEV=0   - Intel 82801AA-ICH, Intel 82801AA-ICH - Direct sample mixing device
  dmix:CARD=I82801AAICH,DEV=1   - Intel 82801AA-ICH, Intel 82801AA-ICH - MIC ADC - Direct sample mixing device
  dsnoop:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - Direct sample snooping device
  dsnoop:CARD=I82801AAICH,DEV=1 - Intel 82801AA-ICH, Intel 82801AA-ICH - MIC ADC - Direct sample snooping device
  hw:CARD=I82801AAICH,DEV=0     - Intel 82801AA-ICH, Intel 82801AA-ICH - Direct hardware device without any conversions
  hw:CARD=I82801AAICH,DEV=1     - Intel 82801AA-ICH, Intel 82801AA-ICH - MIC ADC - Direct hardware device without any conversions
  plughw:CARD=I82801AAICH,DEV=0 - Intel 82801AA-ICH, Intel 82801AA-ICH - Hardware device with all software conversions
  plughw:CARD=I82801AAICH,DEV=1 - Intel 82801AA-ICH, Intel 82801AA-ICH - MIC ADC - Hardware device with all software conversions

graham@graham-xubuntu:~$
```

6 Playing the music

To get playing music, open up your web browser and point it again at port 9000 on the IP address of your server. In the top-right, next to the Logitech logo, there's a small drop-down menu and you'll be able to select your *Squeezelite* player from here. You can add players for different rooms and control them all from here. With a player selected, use the left panel to navigate through your music and either add tracks to the dynamic playlist or play whole albums. You can play music from your other music sources in the same way, and as soon as you click Play, the *Squeezelite* client will make a sound – if MP3 sources don't work, make sure you installed the **libmad0** or **mpg123** libraries. You can remotely control the volume from the web browser and we'd also recommend installing the free *Squeezebox* app in Android for complete remote control. LV



All Songs
AFX - Hangable Auto Bulb EP - 01 - Children Talking.mp3
AFX - Hangable Auto Bulb EP - 02 - Hangable Auto Bulb.mp3
AFX - Hangable Auto Bulb EP - 03 - Laughable Butane Bob.mp3
AFX - Hangable Auto Bulb EP - 04 - Bit.mp3
AFX - Hangable Auto Bulb EP - 05 - Custodian Discount.mp3
AFX - Hangable Auto Bulb EP - 06 - Wabby Legs.mp3

LINUX VOICE TUTORIAL

CREATING AN ANDROID APP WITH APP INVENTOR 2

LES POUNDER

Did you think that building your first Android app world be difficult? Let us guide you through a tool that's as easy as Scratch.

WHY DO THIS?
- We will learn the basics of building an application using *MIT App Inventor 2* and how we can use coding concepts within our application.

TOOLS REQUIRED
- A computer running a recent version of *Firefox* or *Chrome/Chromium*.
- An Android device with a camera.
- An internet connection.
- A barcode scanning app installed on your Android device. We used Barcode Scanner by ZXing Team. **https://play.google.com/store/apps/details?id=com.google.zxing.client.android**

Since the introduction of smartphones in the mid 2000s the world has been gripped by the latest apps. From *Angry Birds* to lifestyle guides the app has become part of our daily lives. Typically an application for the Android operating system is written using an application such as *Android Studio*, which uses a traditional textual language that closely resembles Java. For some this may prove daunting as the editor is rather a large beast to deal with. So how can we enable children to learn the basics of creating an app? Well by following this tutorial of course.

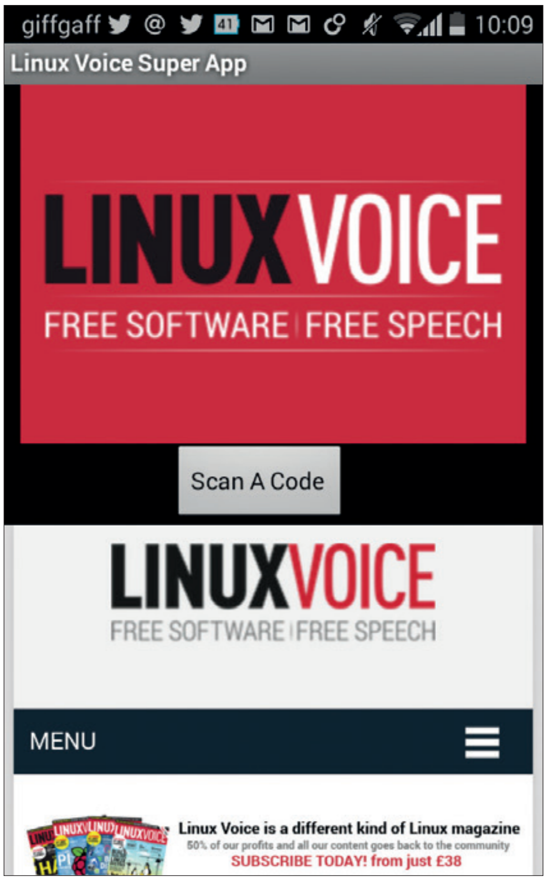
In this tutorial we will be using an online editor called *MIT App Inventor 2*, which enables anyone to create their own Android application using an interface that's not too different to Scratch.

First, open up a web browser and visit **http://appinventor.mit.edu** then click on Create, which is in the top-right of the screen. In order to use *App Inventor* you will need to register for an account; this enables you to create and store your projects in the cloud enabling access from another machine.

The first interface that you can see is the Designer interface, and in here you will create the look and feel for your application, for example adding images, text and buttons. The Designer interface is split into four panes, and they are from left to right:

Palette

Just like Scratch has a palette of commands, so does *App Inventor*. In here you'll find user interface components such as buttons, lists and picker applets. There are also components for layout, working with media such as audio and video, canvas tools to create graphics. You can also utilise sensors such as location, barcodes and accelerometer using this tool. In the Social menu you have tools for accessing contacts, sending and receiving text messages and even Twitter. In the Storage menu you'll find different storage formats for your projects data. In the last two menus, Connectivity and Lego Mindstorms, you'll find components that enable your device to talk over Bluetooth and work with Lego Mindstorms devices.

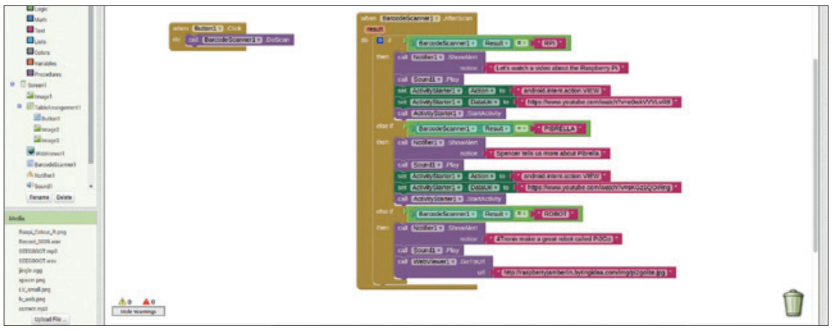

giffgaff 10:09
Linux Voice Super App
LINUX VOICE
FREE SOFTWARE | FREE SPEECH
Scan A Code
LINUX VOICE
FREE SOFTWARE | FREE SPEECH
MENU
Linux Voice is a different kind of Linux magazine
50% of our profits and all our content goes back to the community
SUBSCRIBE TODAY! from just £38

Our application is a QR code scanner app that triggers the playback of multimedia content.

Viewer

Components from the palette can be dragged into the centre of the screen, the Viewer, where there's a simulation of a phone screen. Components such as buttons, lists and images are known as visible components, in that they can be seen on screen. Non-visible components such as TextToSpeech or Sound can be controlled using visible components.

To the right of the Android device simulation we can see the Components pane, which shows all of the components that are in use for our project. By clicking on a component the final pane, labelled Properties, changes focus to reflect the component that has



```
when Button1 .Click
do  call BarcodeScanner1 .DoScan

when BarcodeScanner1 .AfterScan
 result
do  if  BarcodeScanner1 . Result = "RPI"
    then  call Notifier1 .ShowAlert
                 notice "Let's watch a video about the Raspberry Pi"
          call Sound1 .Play
          set ActivityStarter1 . Action to "android.intent.action.VIEW"
          set ActivityStarter1 . DataUri to "[illegible]"
          call ActivityStarter1 .StartActivity
    else if  BarcodeScanner1 . Result = "PIBRELLA"
    then  call Notifier1 .ShowAlert
                 notice "Spencer tells us more about Pibrella"
          call Sound1 .Play
          set ActivityStarter1 . Action to "android.intent.action.VIEW"
          set ActivityStarter1 . DataUri to "[illegible]"
          call ActivityStarter1 .StartActivity
    else if  BarcodeScanner1 . Result = "ROBOT"
    then  call Notifier1 .ShowAlert
                 notice "[illegible] make a great robot called [illegible]"
          call Sound1 .Play
          call WebViewer1 .GoToUrl
                 url "[illegible]"
```

The Designer interface contains all of the components that are used to interact with the application and other forms of input, such as sensors and cameras.

been selected. In the Properties pane we can alter various aspects of components.

So where the Designer interface provides a means to alter the look of a project, our next interface will enable us to program the behaviour of our project.

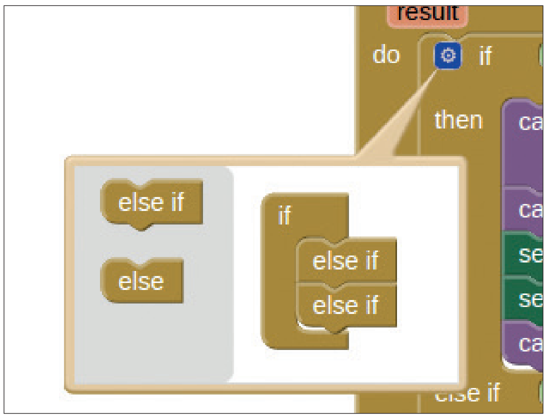
Our project

In our project we will be creating an application that can read QR codes and use the data contained in the code to trigger the playback of audio, video and images. Potential use cases for this type of project can be interactive art installations, notice boards in schools and public places and audio guides for working with equipment in a makerspace.

The project works like this.

```
User presses button.
    Button is programmed to launch a barcode scanner app

A QR code is scanned and the value contained in the code is translated.
The value is then compared to three known values using If...Else If...Else If.
If the value is equal to "RPI".
    Create a pop up window to display a message.
    Play audio.
    Play a Youtube video.

Else If the value is equal to "Pibrella".
    Create a pop up window to display a message.
    Play audio.
    Play a Youtube video.

Else If the value is equal to "ROBOT"
    Create a pop up window to display a message.
    Play audio.
    Open an image on your phone.
```

We'll start by creating a new project, so make sure that you're in the Designer interface. You can create a new project via the Projects drop-down at the top-left of the screen. Select Start New Project and give your project a name followed by pressing OK to continue. After a few seconds the screen will update and present a blank representation of an Android device.



Initially the **if** construction only has one possible condition. You can add more using **else if**, which can be found via the blue cog icon.

Running and installing apps

MIT App Inventor 2 comes with two solutions to test your apps. The first is a companion app that can be installed via the Play Store, it's called MIT AI2 Companion and it is a free download. This app connects your Android device to your app and enables you to test and amend your app, with changes being instantly reflected on your Android device. This is the best way to test your app as you will be testing on real hardware. For this tutorial I tested compatibility using two devices, an HTC Desire HD running Android 2.3 and a Motorola Moto G running Android 4.4.

If you do not have access to an Android device then you can use the Android Emulator which can be found via their website **http://appinventor.mit.edu/explore/ai2/linux.html**

Installation of the emulator for Debian based systems is made easier thanks to a dedicated DEB file. For other Linux distros download the tar.gz archive and follow the instructions for installation.

Once installed the emulator should be called from the terminal using the following command

```
/usr/google/appinventor/commands-for-Appinventor/aiStarter &
```

The emulator will now work in the background, return back to the App Inventor interface and click on Connect > Emulator, found at the top centre on the screen. This will now enable your project to connect to the Emulator and run a virtual Android device. On the home screen of the device is the MIT AI2 Companion app. Use your mouse to activate the Companion app, and in a few seconds your project will be on the virtual Android device.

For our project we will need the following visible components

- **Image x 3**
- **TableArrangement**
- **Button**
- **WebViewer**

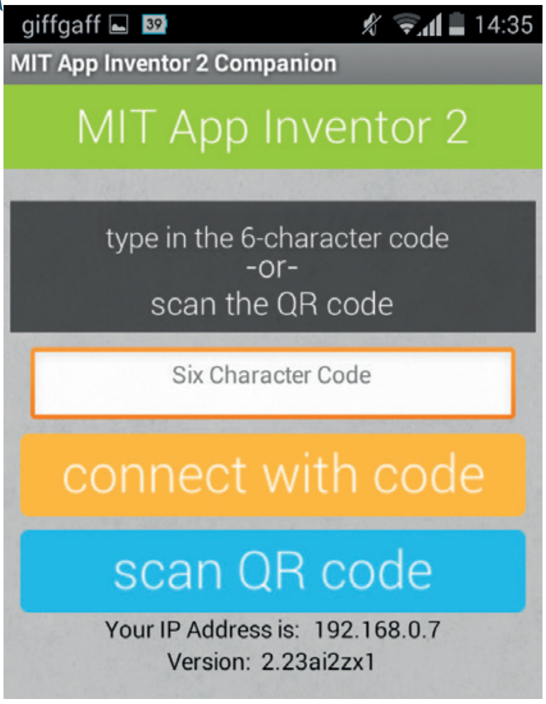
And the following non-visible components

- **BarcodeScanner**
- **Notifier**
- **Sound**
- **ActivityStarter**

Visible components

From the User Interface section of the Palette we will grab the image component and drag it into the Viewer area, ensuring that the component is hovering over the Android screen. Let go of the image component and it will snap to the top-left of the screen. This will be the image for the top of our app. If you would like the image to stretch across the screen look in the Properties pane for the width option and change it to Fill Parent. With the image in place our next component is an invisible TableArrangement component that can be found in the Layout section. Place this underneath the image and change its properties so that it has one row and three columns, as this will help us later in the project. Next, insert a Button component into the middle cell of the table that you have just created, and change the text property of the button so that it instructs the user to press it.

Next we need to do a little hacking to ensure that our button is centred. The best way to do this is to create an image that is 100 pixels wide by 10 tall using the *Gimp* image editor, upload it to *App Inventor* and place it in each of the cells to the left and right of the button using Image components. For each image edit its properties so that it Fills Parent, and hey presto your button will be centred.



Your IP Address is: 192.168.0.7
Version: 2.23ai2zx1

The companion app, which enables you to test and debug your application before installing it, can be found in the Google Play store.

For our last visible component we'll use a WebViewer from the User Interface section. The WebViewer enables web content to be displayed inside an app. We changed the HomeUrl property to the Linux Voice website.

Non-visible components

In the Sensors section we can see the BarcodeScanner component; drag this on to the Viewer pane. The component will not rest in the Android device, rather it will drop to the bottom of the Viewer pane due to it being a non-visible component. From the User Interface section grab the Notifier component and drop it into the Viewer. Our next component is the Sound Player, which can be found in the Media section. Drag this in the same way as the other components. With the Sound Player component highlighted you'll see the Properties panel change to reflect the properties of this component. In this case we see the Minimum Interval and the Source; we are interested in the Source. Click on the white box under the source and a simple menu will pop up. Click on "Upload File..." and select the audio file from your computer. *MIT App Inventor* is compatible with WAV files and MP3. The last component to add to the app is ActivityStarter, which can be found in the Connectivity section. This component enables us to open other applications with our app, for example YouTube or Google Maps.

With the design of our app complete for now, let's turn our attention to creating the code that will make our app come to life. Change to the Blocks interface via the button in the top-right of the screen. In the Blocks interface we can see the blocks that we can use in our project – these comprise the built-in programming concepts that are specific to the components used in the Designer interface.

We'll start with creating the Button Clicked event. In the Blocks pane find Button 1(it will be under Screen 1). Click on Button 1 and a new menu full of blocks will appear. We're interested in the top block, which is a C-shaped block labelled "When Button1.Click". Click and drag the block on to the viewer. C-shaped blocks such as this are used for control and events, such as pressing a button. Inside the "When Button1.Click" block we need to add "call BarcodeScanner1.DoScan" which is found inside the BarcodeScanner1 blocks menu. So what have we just achieved? We have created an event (the button press), and an output that is triggered by the event (launching the barcode scanner app).

Evaluate the QR code

With the barcode-scanning aspect of the project complete, we need to build an algorithm that evaluates the code presented and acts accordingly. When the barcode scanner reads a QR code, the code contained within is decoded into plain text. This is then compared to the hard-coded values in our algorithm. To start we need another C-shaped block from BarcodeScanner1 labelled "When BarcodeScanner1.AfterScan" and inside it we will store the algorithm to run after a successful scan. To start our algorithm open the Control Blocks and locate "If...Then" and drag it into the Viewer pane. You will notice that there's only room for one condition and the resulting output triggered by it. For our algorithm we need to have two further conditions to evaluate against and to add them we need to click on the small blue icon located in the top-left of the C-shaped block. This will create a pop-up window that enables us to alter the structure of the block to accept further conditions by dragging the else if into the smaller representation of the algorithm, once completed you must click on another part of the screen to close the small pop up. Our C-shaped If block now has two further conditional checks that we can now use in our code. Inside the first of our conditions, "If", we set the first test

If the results of the scan are the same as "RPI"

To create this test we need to use a number of blocks. First, look inside the Logic blocks for '**_ = _**'. This block uses comparative logic to evaluate the left value to the right. If both are the same, the answer will be True, and the condition is met. In the first blank we need to place the "BarcodeScanner1.Result" from the BarcodeScanner1 Blocks and in the second blank we need to use a block from Text, which is just a blank

Project code

You can find the complete code for this project at our GitHub repository **https://github.com/lesp/LinuxVoice_Issue13_AppInventor**, or those of you unfamiliar with Git can download the complete package as a ZIP file from **https://github.com/lesp/LinuxVoice_Issue13_AppInventor/archive/master.zip**.

QR codes

QR codes have been with us for many years and have been used to create an automated method of launching a web browser to a specific page and been used to automatically send an SMS to a specific number.

QR stands for Quick Response and was developed in 1994 by Denso Wave as a means to track vehicles during the manufacture process for the Japanese motor industry.

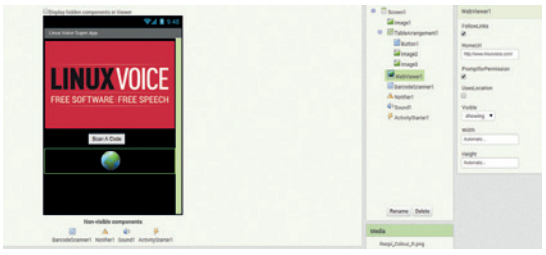
In this project we created a series of QR codes that stored a plain text value that was passed to our app once the code was scanned. To create your own QR codes there are many online resources; for this project we used the resources at **http://qrcode.kaywa.com**, which are free of charge. To use the site, select the type of code that you require, in this case text, which can be found under the More drop-down menu. In the white box type in the text – we used RPI for our first code – then click Generate. Hey presto you have created a QR code.

string. Inside that string we need to type in the text that will be contained in our QR code. With the first condition created now we need to create the code that will run once it is met. We'd like to have a pop-up message appear on the screen. For this we can find the block in Notifier1 called "call Notifier1.ShowAlert notice". Attach this block to the "then" section of the If condition. Next we need to use a Text block labelled " " (for a blank string) and attach it to the ShowAlert block that we have just placed – remember to add your own text to the blank string. This now creates a pop-up that will produce a message once a known code has been scanned. Now we need to add the "Call Sound1.Play" block from Sound1 and attach it to the pop-up message that we just created. This will play the audio file that we earlier uploaded.

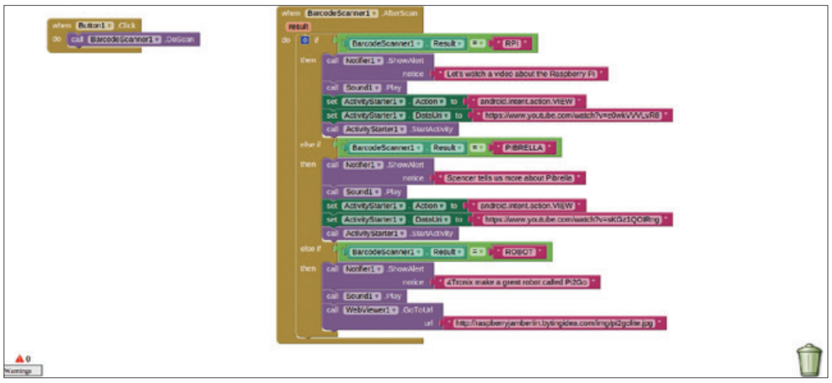
Our next series of blocks inside of the If condition control opening an external application, in this case YouTube. From the ActivityStarter1 blocks use "set ActivityStarter1.Action to" and also grab a blank string block from Text. Inside the blank string box, type the following.

```
android.intent.action.VIEW
```

This instructs the app to open an external viewer application. Our next block is also from the ActivityStarter1 palette and is "set ActivityStarter1.DataUri to". Again, you will need a blank string block from the Text palette. Inside the blank string block, paste in the link to a YouTube video. Our last block in the If condition is "Call ActivityStarter1.StartActivity" and this block will start the process of calling the



The final application should look something similar to this layout which we used in this tutorial.



```
when Button1.Click
do  call BarcodeScanner1.DoScan

when BarcodeScanner1.AfterScan
  result
do  if   BarcodeScanner1.Result = "RPI"
    then call Notifier1.ShowAlert
              notice "Let's watch a video about the Raspberry Pi"
         call Sound1.Play
         set ActivityStarter1.Action to "android.intent.action.VIEW"
         set ActivityStarter1.DataUri to "https://www.youtube.com/watch?v=c0wkVVVLvR8"
         call ActivityStarter1.StartActivity
    else if BarcodeScanner1.Result = "PIBRELLA"
    then call Notifier1.ShowAlert
              notice "Spencer tells us more about Pibrella"
         call Sound1.Play
         set ActivityStarter1.Action to "android.intent.action.VIEW"
         set ActivityStarter1.DataUri to "https://www.youtube.com/watch?v=sKGz1QO8Rng"
         call ActivityStarter1.StartActivity
    else if BarcodeScanner1.Result = "ROBOT"
    then call Notifier1.ShowAlert
              notice "4Tronix make a great robot called Pi2Go"
         call Sound1.Play
         call WebViewer1.GoToUrl
              url "http://raspberryjamberlin.bytingidea.com/img/pi2golite.jpg"
```

The blocks used to program your application should end up looking like this. A high resolution version can be found in the Github repository for this project.

external viewer application, in this case YouTube, and pass it the URL for the video.

For the next "Else If" condition we can duplicate the code already written for "If" by right-clicking on the code and selecting "Duplicate"; once you have the duplicate blocks, arrange them inside of each "Else If" section just like before. Remember to change the text that we expect to see in the QR code, and the message contained in the pop up.

For the last "Else If" condition the blocks contained are a little different. The condition test, comparing the QR code to a keyword is the same, but the code that is run once this condition is true is different. A pop up is still created and a sound is played, but rather than call a YouTube video the app will open an image inside the app. To do this grab "call WebViewer1.GoToUrl" from WebViewer1 palette, and grab a blank string from Text. Inside the blank string block paste a link to an image. This action will trigger an image to appear inside of our app. Well done: you've made an app!

To test your app on your Android device you'll need to connect your device to the internet and then start the *MIT AI2 Companion* app on your Android device. On your PC in the Blocks or Designer interface, click on Connect and then AI Companion. You can enter the code from the PC into the Companion app on your device or you can scan the QR code. The two should now connect up and your app will be on the screen of your device. Once you're happy with your app you can build it into an installable file for use on many devices. To do this, navigate to the Build menu and select "App (provide QR code for .apk)". This will build the application on MIT's servers and provide you with a QR code to download the app to your Android device. Use the barcode scanner app installed on your device to download the file.

Once downloaded you can simply click on the file to install it on to your device. Typically apps are installed from a "trusted" source, which is the Play store. Installing your own apps requires that you allow installation from "Unknown Sources", and you can find this option in your device's Settings > Security. Put a tick in the box and retry installation of your app.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

LINUX VOICE TUTORIAL

DDRESCUE: SALVAGE DATA FROM DAMAGED DISKS

MARK CRUTCH

How a GNU and a penguin rescued a bear from a broken hard drive and the clutches of the evil empire.

WHY DO THIS?

- Earn respect and admiration by recovering 'lost' data.
- Learn about disk images.
- Remind yourself that it's time to make another backup!

LV PRO TIP

Some distros' repositories contain both **dd_rescue** and **GNU ddrescue**, so double-check which one you're installing. Debian derivatives use **gddrescue** as the package name for the GNU version and **ddrescue** for Kurt Garloff's **dd_rescue** command.

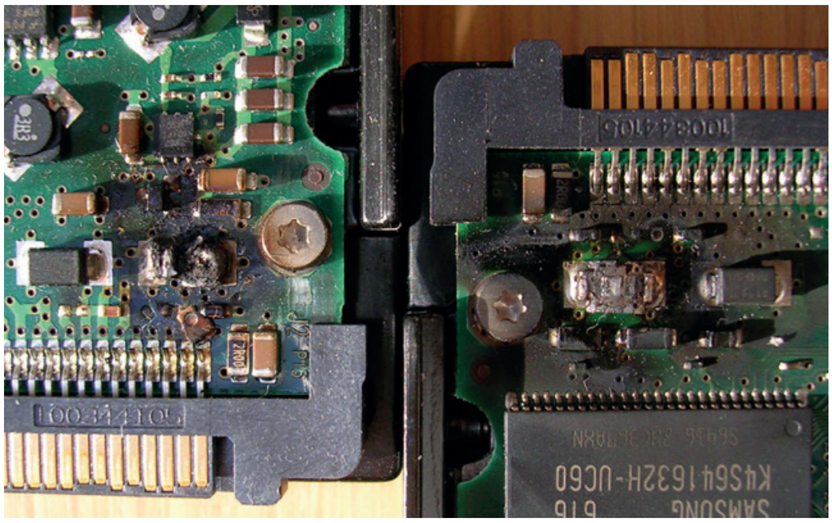
Come with us, gentle reader, on an exciting adventure into the world of data recovery. There will be loss and sadness, a hostage-taking mega corporation, a triumphant recovery, and lessons learned. Plus a comic book bear with an eye-patch. Welcome to the first draft of *Bertie Bear and the Disk of Corruption*...

Chapter 1: An artist's grief

It seemed like any other day for independent artist Andy Clift, as he booted his Apple Mac to continue work on the latest instalment of his comic book series, "Bertie Bear and the Dagger of a Thousand Souls". But within seconds of hearing the familiar start-up chime, his Apple turned sour. "Drive Error", it reported, refusing to proceed any further. Andy looked on in dismay, his mind churning with frantic thoughts; he knew he had backups of most of his files, but the latest drawings of Bertie's adventures had yet to make their way off the reluctant drive. He packed his iMac into a box and headed off to the nearest Apple store's "Genius Bar".

Too late he found that the "Genius Bar" is something of a misnomer. The staff there are better trained than most sales assistants, but it transpires that there's not even an IQ test required, let alone membership of Mensa, before a keen employee can be promoted to the position. Suffice to say that the "genius" failed to demonstrate any advanced skills beyond the ability to send the machine to Apple's service centre for a new drive to be fitted. Resigned to never seeing his latest creations again, Andy made his way to one of the mockingly intact Macs in Apple's store, and proceeded to post to his blog.

The scorch marks show where a single bad PSU simultaneously destroyed the electronics of both these drives.



Chapter 2: A drive held hostage

Meanwhile, at Linux Voice HQ, a Raspberry Pi was grepping its way through the internet. It paused to parse Andy's message, before triggering the launch of a foam dart and the careering dance of an ice-cream tub with wheels. This was our signal to leap tigerish into action.

"Somebody on the internet's got a hard drive problem!" cried our illustrious leader, his fingers already walking over to a pile of Linux CDs as he contemplated his rescue plan.

"But it's only a Mac user," replied a cynical voice from the corner, rising on a column of solder fumes and flux.

"We're better than that!" came the response. "We should be prepared to help our fellow man wherever we can. Besides, it's a good opportunity to see how effective Linux's HFS+ support is."

So we contacted Andy to offer our meagre skills in trying to recover his work from the drive, just as soon as he received it back from Apple.

You might expect that, having paid Apple over £200 to replace the hard drive, the old disk would be returned with the refurbished machine. Instead Apple demanded a ransom (our term, not theirs) of an extra £90 for its return! At first Andy was reluctant to increase his spend to almost £300, but when we pointed out that his credit card details, usernames and passwords would soon be in the hands of some third party salvage company, he decided to pay the ransom and reclaim the drive. Any data we could get off it would be a bonus.

When the disk arrived our first step was a visual inspection. We've seen several drives rendered useless by bad power supplies, but with no obvious burn marks or charred components on the visible side of the drive's circuit board, we connected it to our recovery machine. Attaching it straight to the motherboard would give us the fastest transfer speeds, but unseen electrical problems would be more likely to damage the host machine. We chose, instead, to place it into an external drive caddy, to provide a little extra electrical insulation, at the expense of limiting the data transfer to USB2 speeds.

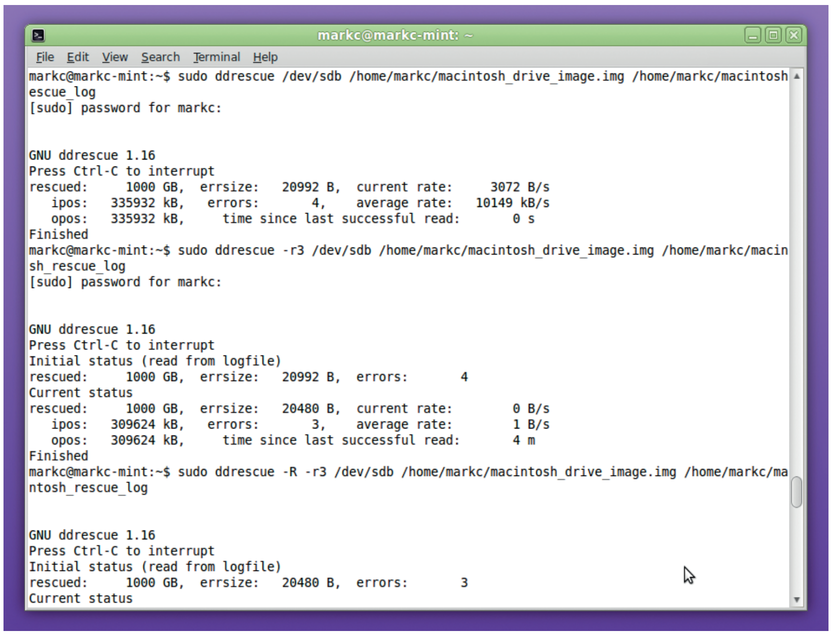
We plugged in the USB connector and to our delight the drive was picked up instantly, our Mate desktop promptly opening a window showing the drive's contents. This told us that the device was basically working, and we listened for the tell-tale sounds of mechanical issues emanating from the disk's moving components. We also took the opportunity to examine the structure of the drive using Mate's 'Disks' program and found that there were three partitions: 'boot', 'data' and 'recovery'.

With everything appearing – and sounding – as a good drive should, we guessed that the problem was down to a few bad sectors that had been enough to annoy OS X. With no idea where on the disk those bad sectors might lie, it would be foolish to simply copy the data using the normal desktop tools, so we opted to create a full disk image to work with. Now we ask you, honoured reader, to imagine the strumming of a harp and a wobbly fade effect, as we take a break in our narrative to switch to the flashback section...

Chapter 3: A tale of four DDs

As the soft sound of the harp dwindles to nothing, we find ourselves at the dawn of time. Okay, a little after the dawn of time, but still pretty early in the annals of history. And, of course, we mean shortly after the dawn of "Unix time" – 1 January 1970.

Back in those early days of Unix the **dd** command was created as a means of copying blocks of data between devices. You can read more about it in Linux Voice #08, but it falls into our story because it's a classic method for cloning from a drive to an image file. Because **dd** deals with blocks of data directly, it can be used to create a sector-for-sector clone of a drive even if it has foreign partitions. Unfortunately the way **dd** works makes it less than ideal for recovery tasks: it aborts on read errors, for example, which is definitely not what you want when dealing with a suspect drive.



```
markc@markc-mint:~$ sudo ddrescue /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/macintosh
escue_log
[sudo] password for markc:


GNU ddrescue 1.16
Press Ctrl-C to interrupt
rescued:     1000 GB,  errsize:   20992 B,  current rate:     3072 B/s
   ipos:   335932 kB,   errors:       4,    average rate:   10149 kB/s
   opos:   335932 kB,     time since last successful read:       0 s
Finished
markc@markc-mint:~$ sudo ddrescue -r3 /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/macin
sh_rescue_log
[sudo] password for markc:


GNU ddrescue 1.16
Press Ctrl-C to interrupt
Initial status (read from logfile)
rescued:     1000 GB,  errsize:   20992 B,  errors:       4
Current status
rescued:     1000 GB,  errsize:   20480 B,  current rate:        0 B/s
   ipos:   309624 kB,   errors:       3,    average rate:        1 B/s
   opos:   309624 kB,     time since last successful read:       4 m
Finished
markc@markc-mint:~$ sudo ddrescue -R -r3 /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/ma
ntosh_rescue_log


GNU ddrescue 1.16
Press Ctrl-C to interrupt
Initial status (read from logfile)
rescued:     1000 GB,  errsize:   20480 B,  errors:       3
Current status
```

The first pass took 28 hours. Thankfully the subsequent runs only took minutes.

Years later, building on the name and basic premise of **dd**, Kurt Garloff created **dd_rescue**, a tool specifically designed to recover data from failing drives. It has error handling that enables it to keep going where **dd** would fail. But that also means that it can take a very long time to image a drive with lots of read errors. To speed up this process, Valentin Lab created a *Bash* script called **dd_rhelp**, which optimises the way in which **dd_rescue** performs its job. When **dd_rescue** finds errors **dd_rhelp** makes it re-start at a later sector, hoping to find another good section of the drive, while keeping a log of the recovered parts of the drive so that it can work out which ones still need to be revisited. In this way it aims to recover the readable parts of the drive as quickly as possible, before going back to areas that may not yield any useful results.

"dd can be used to create a sector-for-sector clone of a drive even if it has foreign partitions."

It seemed that the combination of **dd_rescue** and **dd_rhelp** is just what we needed, but there was one more contender to consider: the confusingly named **ddrescue** (without the underscore).

Officially known as "GNU ddrescue", **ddrescue** aims to do the job of the **dd_rescue**/**dd_rhelp** combination, but in a single application. It keeps a log file of its progress, and attempts to speed through the readable data on a drive as quickly as possible, coming back to bad areas later. It can be stopped and then resumed at another time, and you can run it repeatedly without affecting previously recovered blocks. On our Linux Mint box, **sudo apt-get install gddrescue** was the right invocation to install it. With our weapon of choice in place, it's time to return to our adventure. Cue harp and wobbly fade as we head back to the present day...

Alternative tools

During this little adventure we used the Mate *Disks* application. This was formerly known as *Gnome Disk Utility* or *Palimpsest*, and is available in most Gnome-derived desktop environments. If you're using a different environment you can still do everything in the article, but you'll need an alternative set of tools.

For simply looking at the partition structure of a disk or mounted image, the venerable **fdisk** command line tool is present on almost all distributions, as is the GNU **parted** application. If you prefer a GUI then *GParted* is a great GTK front-end to **parted**, or you might prefer the *KDE Partition Manager*, which also uses GNU's **libparted** library under the hood.

Mounting a partition within a full disk image is a little more tricky, with a general lack of GUI tools. It is possible to manually calculate the start position of your partition, then use it as an offset to the **mount** command. A less masochistic option is the **kpartx** utility, which can list partitions in a disk image and mount them all for you, without the need for maths.

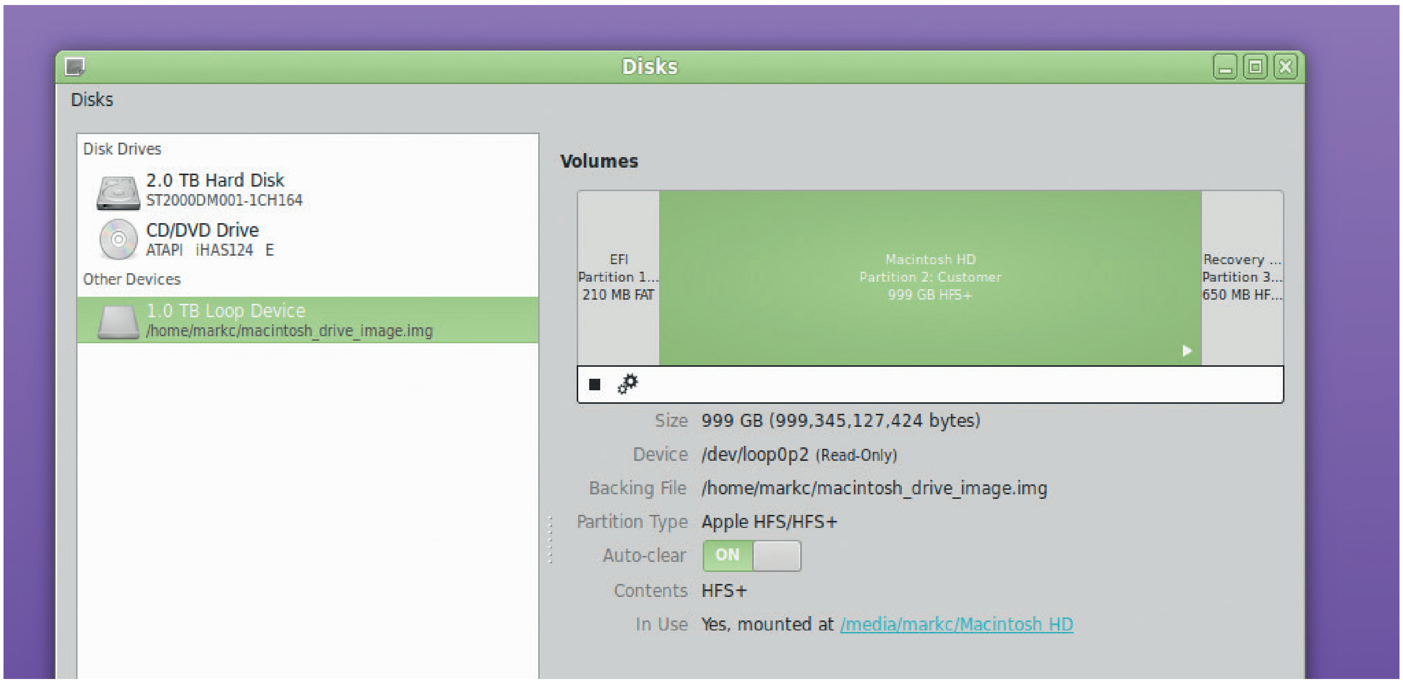
```
kpartx -av hard_drive.img
```

This will mount all the partitions to devices under **/dev/mapper** in the filesystem – see the **kpartx** man page for more details. Once you're done remember to delete the partition mapping and unmount the image:

```
kpartx -d hard_drive.img
```

LV PRO TIP

Use Ctrl+C to stop **ddrescue**. Provided you use a log file you can just start it again later and it will resume from where it left off. This even works if your machine crashes mid-recovery!


Disks
Disks
Disk Drives
2.0 TB Hard Disk
ST2000DM001-1CH164
CD/DVD Drive
ATAPI iHAS124 E
Other Devices
1.0 TB Loop Device
/home/markc/macintosh_drive_image.img
Volumes
EFI Partition 1... 210 MB FAT
Macintosh HD Partition 2: Customer 999 GB HFS+
Recovery ... Partition 3... 650 MB HF...

Size 999 GB (999,345,127,424 bytes)
Device /dev/loop0p2 (Read-Only)
Backing File /home/markc/macintosh_drive_image.img
Partition Type Apple HFS/HFS+
Auto-clear ON
Contents HFS+
In Use Yes, mounted at /media/markc/Macintosh HD

Yes, that bit of text in the corner that says "Disks" is actually a menu.

Chapter 4: An ursine rescue

With Andy's drive showing up as **/dev/sdb** on our penguin-powered machine it was time to send **ddrescue** stampeding in to flush out the easily recoverable data. We just had to specify the source drive followed by the names of the image and log files. Note that the command is **ddrescue**, even though the package we installed was **gddrescue**.

```
sudo ddrescue /dev/sdb hard_drive.img rescue_log
```

To our dismay the first errors arrived quickly. The errsize figure in the output grew rapidly. 40,000 bytes... 50,000... then at just over 60kB the figure stopped increasing. Could we really have been so lucky? Was the damaged data confined solely to the boot partition, meaning that Andy's personal files were all intact? We wouldn't know the answer for some time – over 28 hours at USB2 speed – when **ddrescue** finished its first pass.

The errsize still stood at 60kB!

We didn't want anything from the boot partition, so there was no real need to continue. But we were curious to discover just how much **ddrescue** might be able to recover from those damaged sectors. We let it run on, continuing through its remaining phases, and quickly the errsize dropped to about 20kB.

20kB of bad data after just a single run was certainly impressive. But stubborn sectors can sometimes be persuaded to give up their data if you just ask them often enough, so we ran **ddrescue** a second time, instructing it to retry each bad sector up to three times.

```
sudo ddrescue -r3 /dev/sdb hard_drive.img rescue_log
```

That recovered another half a kilobyte of data. Perhaps we could surprise the drive into responding by sneaking up on it from the other direction? Adding **-R** to the command told it to read the sectors in reverse order, working back into the damaged areas.

```
sudo ddrescue -R -r3 /dev/sdb hard_drive.img rescue_log
```

Almost 15kB was recovered by that approach, leaving us with only 5,120 bytes of unreadable data. We tried a few more passes, but no additional data was forthcoming. Still, 5kB of bad data seemed pretty good to us – and as it was all on the boot partition we were confident that Bertie Bear would live to fight another day. Had the errors been on the data partition then we might have persevered a little more. With any suspect drive, however, there's always a danger that you'll speed up the degradation of the device, so the rule should usually be to get as much data as possible, as quickly as you can, and only spend extra time on stubborn sectors if you really need to.

LV PRO TIP

Sometimes a damaged drive is more cooperative when it's had a chance to cool down. Stop **ddrescue**, detach the drive for a while, then reattach and resume a little later.

LV PRO TIP

dd, **dd_rescue** and **ddrescue** all use different command line options and parameters, so make sure you know which one you're using, as a mistake could be disastrous.

Chapter 5: A bear in a gilded cage

Although we now had an image of the drive to work with, in some respects we'd actually taken a step backwards. Whereas we had previously been able to access the files on the drive directly from the desktop, now the crown jewels we sought were trapped inside a partition which was in turn inside a disk image.

We were only really interested in one of the three partitions. Had we imaged each one individually we would be able to mount it directly using Linux's loopback interface. But we'd imaged a whole drive, with partitions inside it. We needed a way to tell Linux to mount the drive, then mount the partitions within it, before we could gain access to the files themselves.

It turns out that Mate's *Disks* application has a secret ability. It does such a god job of looking like a simple, single dialog application that few people notice it has a menu bar, hiding in plain sight. Clicking on the lone menu reveals that it holds an entry that reads "Attach Disk Image".

Using that option to attach our disk image immediately placed it into the list of drives alongside

When recovery gets tough

We were lucky with this recovery job because the damaged sectors were all in an unneeded partition. But what would we have done if the damage had been to a partition we wanted? In any data recovery situation you should always make a disk image first, rather than working directly on the suspect drive, so the **ddrescue** step would be similar – but we would probably have been a little more careful in our subsequent runs to recover as much as possible. The most likely result would be a readable disk image with some missing data. In that case it's simply time to keep your fingers crossed that the rogue bytes aren't in any files you actually want.

If the partition information itself is unrecoverable it's time to install *TestDisk* and *PhotoRec*, a pair of applications written by Christophe Grenier. These are often bundled together: installing them both on a Debian-based system just requires a single **sudo apt-get install testdisk** command.

TestDisk (though the executable name is **testdisk**) can be used to recover lost and damaged partitions by analysing the disk structure and recognising a number of partition types. If *TestDisk* is unable to recover the partition, *PhotoRec* can often recover files at an even lower level. Despite its name, *PhotoRec* can find more than just photos: it understands an extensive list of file types, and it's possible to add your own file signatures should you need something more esoteric. It works by reading blocks from the disk or image directly, so can recover files even if the partition format is unknown. *PhotoRec* only works reliably on unfragmented data, though, so don't expect miracles when dealing with a well-used drive that's full to bursting.

You can find out more about *TestDisk* and *PhotoRec*, including worked examples, at Christophe's website: **www.cgsecurity.org**.

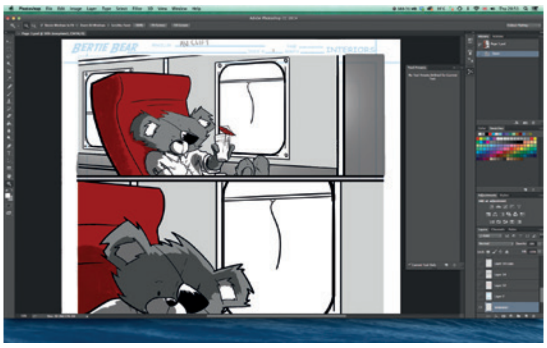
our other, physical devices. Selecting it populated the rest of the window with the same overview of the partitions as we had previously seen with the real drive. Then we selected the data partition and clicked the "mount" button. A link appeared, proclaiming the path to the mount point. With some scepticism we clicked the link, paused for a second, then released a sigh of relief as a window opened before us, displaying the contents of Andy's drive in all its Mac-based glory.

You would be forgiven for thinking that the rest was easy. But this is the tale of data recovery across disparate operating systems, and for all the hubris of Silicon Valley the truth is that computers rarely make things that straightforward. Quickly we were stymied by permissions issues preventing us accessing all the files we wanted.

The problem is that Linux's HFS+ support is a little too good. As OS X is a Unix system at heart, so its filesystem carries with it all the finer details of ownership and access rights that you might expect from a Linux-native format. On the Linux box our user ID was 1000. Andy's Mac had given him an ID of 500 – and that ID was gladly honoured by Linux, denying us access to many of the files. In a pique of laziness we used **sudo** to launch Mate's *Caja* file manager, elevating ourselves to a position of computer godhood, so that trivialities like file permissions would no longer impede us. But ask yourself, dear reader, who among you would not have taken the same approach, so long as you thought that nobody was looking?

```
sudo caja --no-desktop
```

The external drive that Andy had sent us was formatted using Microsoft's NTFS filesystem – which doesn't preserve Unix permissions. Knowing that OS X is quite capable of reading from such a drive, we just selected everything in Andy's home directory and dragged it straight to the external drive, assured that the pesky user ID wouldn't be preserved, so wouldn't cause Andy a problem later. Finally Bertie Bear was freed from captivity.



At last Bertie could relax with a drink, now he was back at home in *Photoshop* on Andy's Mac.

Epilogue

A few days later we received news that Andy was able to read the files from the backup drive. Andy's files were intact, and he had learned a vital lesson about making backups. We had discovered a little more about **ddrescue** and how to recover data from inside a partition in a disk image. And we had a rip-roaring adventure to write up for Linux Voice.

Thanks to our efforts, *Bertie Bear and the Dagger of a Thousand Souls, Volume 3* was released on schedule. Interested readers can find this, and the previous two instalments at **http://bertiebear.bigcartel.com**.

But alas! as is so often the case in such tales, the antagonist of our story still lives on and continues with their evil ways. Who knows how many drives are being held hostage by Apple and their ilk? Despite our slight dramatisation it really wasn't too hard to get Andy's personal data from his drive. Imagine what that means for all the "dead" drives that Apple has sent for salvage, or for those that grace the listings of Ebay, that fill the shelves of pawn shops, or that reside in the carcasses of abandoned computers at rubbish dumps across the land. Remember this tale the next time you're tempted to let an old hard drive out of your hands. Oh, and one final thing: go and make a backup. Now. We can't always be there to save you.

Mark Crutch has been helping the world through Linux for a while, but more importantly, he's one half of the team that creates the Elvie cartoon in our letters pages: peppertop.com.

LV PRO TIP

You can use GNU ddrescue as a replacement for dd in a lot of cases – such as writing an OS to an SD card for use in a Raspberry Pi. They will do the same job, but ddrescue provides more feedback as it progresses.

LINUX VOICE TUTORIAL

INTERNET RELAY CHAT: GET TO GRIPS WITH THE IRC PROTOCOL

RICHEY DELANEY

Get to know the IRC protocol, using Telnet and coding a client in Python.

WHY DO THIS?

- Communicate with like-minded people across the internet.
- See inside the workings of a simple IRC client.

For whatever reason, humans seem to want to communicate. Whether it's snapping non-stop selfies or handwriting beautifully crafted manuscripts to send to one another, expressing our opinions remains a staple part of the intellectual diet of most humans. IRC is an open protocol for group communication. It provides a way to transfer simple plain text messages over the TCP protocol.

The beauty of IRC is its simplicity: it's a plain text protocol, which means that you can experience IRC without even having a dedicated client. Later in the tutorial, we'll show how you can use a tool like Telnet to use IRC at the socket level. It has clients on almost every platform imaginable, most with a list of useful plugins.

IRC had its inception in 1988, so it's by no means a new technology. It was born of a need for a better instant communication mechanism when all that was available was bulletin board systems. There are other standards for messaging, both in a group and directly; for example, in the open source world, XMPP is an open standard for real-time communication. Despite these other standards, IRC still holds a special place in the open-source world. Many Linux distributions for example offer a lot of their customer support through IRC, and a lot of development discussion and coordination happens on IRC channels. An example of these would be the **#linuxmint-help** channel on the spotchat IRC network: when a user first installs Linux Mint, the welcome screen has the option of a "chat room" which directs the user to the IRC channel.

IRC is also a great way to get involved and up to speed quickly with an open source initiative. Linux Voice is a good example of this, as most of the magazine founders are on IRC at **#linuxvoice**. It's a great means of sharing feedback and discussing what you enjoyed and perhaps didn't enjoy about the latest issue or podcast. IRC is often a great way to start contributing to an open source project – projects will often have an IRC channel with some of the main developers present who can likely help with any issues you encounter.

Getting started

The easiest way to demonstrate the IRC protocol is using the very simplest of IRC clients, Telnet. Telnet, in case you haven't come across it, is a tool for logging into remote machines similar to SSH. It does not default to a secure connection and therefore these days it is more commonly used as a raw interactive TCP session. This enables the user to type the exact data that they want to send and receive any raw data on the socket. We can use it as a very basic IRC client.

For this example, and for further examples, the IRC network we're using will be **irc.freenode.org** and the port we will be using will be 6667. The port for IRC is officially 194, however as this is a privileged port it means IRC server software cannot be run by anyone other than root. As a result, the unofficial standard port for IRC is 6667.

To connect to the server, run Telnet as follows:

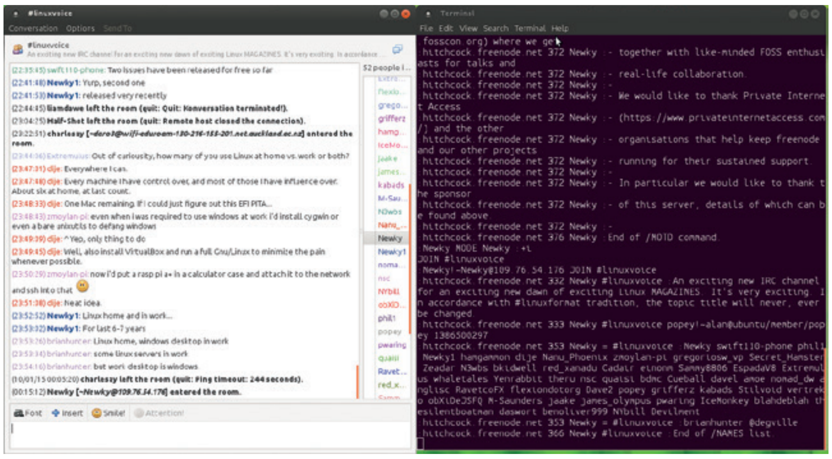
```
$ telnet irc.freenode.org 6667
```

After the connection, you will need to complete the registration phase within a certain time frame or else the server will terminate the connection. If the connection is closed, just start over.

All interactions with the IRC server adhere to a structure. They are plain text messages, and take the following format:

```
[<source>] <command> <parameters> <crlf>
```

Messages being sent from the IRC server to the client will often have a 'source', for example when someone sends a message using the **PRIVMSG** command to a channel of which a user is a member, the source portion of the message will be the user's details. The 'command' will be one of the supported IRC commands. Examples of commands will be seen in the registration phase such as **NICK** and **USER**. Some commands optionally take parameters; for example, the **NICK** command takes the user's nickname as a parameter. The **crlf** in the command above stands for "carriage return line feed" which is how each message is terminated. In some settings,


Go to #linuxvoice, then talk Linux on IRC with *Pidgin* or good old Telnet!

```
#linuxvoice
An exciting new IRC channel for an exciting new dawn of exciting Linux MAGAZINES. It's very exciting. In accordance ...
(22:35:45) swift110-phone: Two issues have been released for free so far
(22:41:48) Newky1: Yurp, second one
(22:41:53) Newky1: released very recently
(22:44:45) liamdawe left the room (quit: Quit: Konversation terminated!).
(23:04:25) Half-Shot left the room (quit: Remote host closed the connection).
(23:22:51) charlesay [~dero3@wifi-eduroam-130-216-155-201.net.auckland.ac.nz] entered the room.
(23:44:36) Extremuius: Out of curiousity, how many of you use Linux at home vs. work or both?
(23:47:31) dije: Everywhere I can.
(23:47:48) dije: Every machine I have control over, and most of those I have influence over. About six at home, at last count.
(23:48:33) dije: One Mac remaining. If I could just figure out this EFI PITA...
(23:48:43) zmoylan-pi: even when i was required to use windows at work i'd install cygwin or even a bare unixutils to defang windows
(23:49:39) dije: ^ Yep, only thing to do
(23:49:45) dije: Well, also install VirtualBox and run a full Gnu/Linux to minimize the pain whenever possible.
(23:50:29) zmoylan-pi: now i'd put a rasp pi a+ in a calculator case and attach it to the network and ssh into that
(23:51:38) dije: Neat idea.
(23:52:52) Newky1: Linux home and in work...
(23:53:32) Newky1: For last 6-7 years
(23:53:26) brianhunter: Linux home, windows desktop in work
(23:53:34) brianhunter: some linux servers in work
(23:54:16) brianhunter: but work desktop is windows
(10/01/15 00:05:20) charlesay left the room (quit: Ping timeout: 244 seconds).
(00:15:12) Newky [~Newky@109.76.54.176] entered the room.
```

```
fosscon.org) where we get
:hitchcock.freenode.net 372 Newky :- together with like-minded FOSS enthusi
asts for talks and
:hitchcock.freenode.net 372 Newky :- real-life collaboration.
:hitchcock.freenode.net 372 Newky :-
:hitchcock.freenode.net 372 Newky :- We would like to thank Private Interne
t Access
:hitchcock.freenode.net 372 Newky :- (https://www.privateinternetaccess.com
/) and the other
:hitchcock.freenode.net 372 Newky :- organisations that help keep freenode
and our other projects
:hitchcock.freenode.net 372 Newky :- running for their sustained support.
:hitchcock.freenode.net 372 Newky :-
:hitchcock.freenode.net 372 Newky :- In particular we would like to thank t
he sponsor
:hitchcock.freenode.net 372 Newky :- of this server, details of which can b
e found above.
:hitchcock.freenode.net 372 Newky :-
:hitchcock.freenode.net 376 Newky :End of /MOTD command.
:Newky MODE Newky :+i
JOIN #linuxvoice
:Newky!~Newky@109.76.54.176 JOIN #linuxvoice
:hitchcock.freenode.net 332 Newky #linuxvoice :An exciting new IRC channel
for an exciting new dawn of exciting Linux MAGAZINES. It's very exciting. I
n accordance with #linuxformat tradition, the topic title will never, ever
be changed.
:hitchcock.freenode.net 333 Newky #linuxvoice popey!~alan@ubuntu/member/pop
ey 1386500297
:hitchcock.freenode.net 353 Newky = #linuxvoice :Newky swift110-phone phil1
 Newky1 hamgammon dije Nanu_Phoenix zmoylan-pi gregoriosw_vp Secret_Hamster
 Zeadar N3wbs bkidwell red_xanadu Cadair einonm Sammy8806 EspadaV8 Extremul
us whaletales Yenrabbit theru nsc quaisi bdmc Cueball davel amoe nomad_dw a
nglisc RavetcoFX flexiondotorg Dave2 popey grifferz kabads Stilvoid vertrek
o obXiDeJSFQ M-Saunders jaake james_olympus pwaring IceMonkey blahdeblah th
esilentboatman daswort benoliver999 NYbill Devilment
:hitchcock.freenode.net 353 Newky = #linuxvoice :brianhunter @degville
:hitchcock.freenode.net 366 Newky #linuxvoice :End of /NAMES list.
```

this is written as **\r\n**.

The registration phase of IRC connection consists of the following:

1. The **NICK** message is sent to register a nickname for the user.
2. The **USER** message is sent to specify the username, hostname and real name of the user.

In this example, the nickname and username "LinuxVoiceTest" and real name "Linux Voice IRC TEST" will be used.

```
$ telnet irc.freenode.org 6667
Trying 185.30.166.38...
Connected to chat.freenode.net.
Escape character is '^]'.
NICK LinuxVoiceTest
USER LinuxVoiceTest 0 * :Linux Voice IRC Test
```

In the **USER** command the first parameter after the command is used to set special user modes, such as making the user invisible by default. For now, we use **0** to log in as a normal user. The IRC defines the third parameter as unused hence the, ***** parameter. The real name is prefaced by a **:** character. Once the **USER** command has been sent successfully, you will get the normal message of the day from the freenode server to signify a successful login.

Congratulations, with these few commands you have officially registered with the freenode IRC server. This illustrates just how simple the protocol is, and how quickly you can get started. If the connection to the server has been left open but inactive for an extended period of time, the server will eventually terminate the connection. Before the connection was disconnected, Telnet will report something like:

```
PING :orwell.freenode.net
:LinuxVoiceTest!~LinuxVoic@<IP ADDRESS> QUIT :Ping timeout:
245 seconds
ERROR :Closing Link: <IP ADDRESS> (Ping timeout: 245
seconds)
Connection closed by foreign host.
```

At certain intervals, the server will send a **PING** to inactive clients to establish whether they should be disconnected. The **PING** will contain some text after it, prefaced with a **:** character. The client must return with a corresponding **PONG** message echoing the same text. In the above case, the connection could have been kept alive by using the **PONG** command.

```
PONG :orwell.freenode.net
```

Now that the normal administration of registering and maintaining an IRC connection is complete, the next step is being able to talk to others on the server. There are a number of different ways to talk on IRC. The two most common of these are private messages, which are one-to-one conversations with other people on the IRC network; and IRC channels, which are group conversations. You can start a channel on freenode simply by using a channel that doesn't currently exist. To get a list of all the available channels and a short description of each, you can use the **list** command with no parameters. Exercise caution with this, as the room list is large for popular servers. For now, we'll join the **#linuxvoice** channel.

```
JOIN #linuxvoice
:LinuxVoiceTest!~LinuxVoic@<IP ADDRESS> JOIN #linuxvoice
:verne.freenode.net 332 LinuxVoiceTest #linuxvoice :An exciting
new IRC channel for an exciting new dawn of exciting Linux
MAGAZINES. It's very exciting. In accordance with #linuxformat
tradition, the topic title will never, ever be changed.
:verne.freenode.net 353 LinuxVoiceTest = #linuxvoice
:LinuxVoiceTest Manj-811-Xfce2 Newky1 ubiquitous1980 huw
hamgammon red_xanadu zmoylan-pi bkidwell Extremulus
RavetcoFX skellat DonOregano1 nomad_dw flexiondotorg Dave2
popey grifferz kabads einonm Zeadar Cadair Stilvoid vertreko
obXiDeJSFQ gregoriosw_vp M-Saunders nsc jaake Secret_
Hamster davel james_olympus pwaring IceMonkey blahdeblah
N3wbs amoe quaisi thesilentboatman bdmc aptanet
Sammy8806 Cueball daswort dizzylizzy theru Devilment
brianhunter
:verne.freenode.net 353 LinuxVoiceTest = #linuxvoice :phil1
Yenrabbi1 @degville NYbill benoliver999
:verne.freenode.net 366 LinuxVoiceTest #linuxvoice :End of /
NAMES list.
```

The server first echoes back the **JOIN** command with the source as our registered user. It is followed by the topic for the channel, followed by the names of each user in the channel already. To find the names of all users in a channel, use **NAMES #linuxvoice**.

"IRC provides a way to transfer simple plain text messages over TCP."

Next, we finally get to share some thoughts with the world. For communicating either privately or with a channel, the **PRIVMSG** command is used. The command specification is simple:

```
PRIVMSG <msgtarget> <text to be sent>
```

As always, the text to be sent should be prefaced with a **:** sign. To send a "Hello, World!" to the #linuxvoice channel:

```
PRIVMSG #linuxvoice :Hello, World!
```

If you are lucky enough that someone says Hi back, you should see something like the following:

```
:Newky1!~<USERNAME>@<IP ADDRESS> PRIVMSG #linuxvoice
:Hello LinuxVoiceTest
```

This message tells us that Newky1 has sent a message to the #linuxvoice channel and with a message text of "Hello LinuxVoiceTest". If Newky1 decides that he wants to talk to us privately and not in front of everyone subscribed to a particular channel, the message target will be our nickname rather than the channel.

```
:Newky1!~<USERNAME>@<IP ADDRESS> PRIVMSG
LinuxVoiceTest :Super secret
```

To reply, the **PRIVMSG** command is used with the message target set to "Newky1".

```
PRIVMSG Newky1 :Super secret reply
```

To leave a channel, we can use the **PART** command with the channel name as the only parameter as follows:

```
PART #linuxvoice
```

You can quit your IRC session altogether by using the **QUIT** command, this will also disconnect you from any channels that you are a part of. Both the **QUIT** and the **PART** commands take a part message (part messages are displayed to others when you leave, for example "Gone to lunch").

Coding a client

Although Telnet is a convenient and simple way to experiment with the protocol, it's clearly not ideal for anything but experimentation. The next section of this tutorial will deal with using the IRC protocol in a Python program. We will build the infrastructure for a simple IRC bot. There are some very sophisticated IRC bots that perform a range of functions including FAQs and helping out new channel members. There are also bots that listen and aggregate links or other statistics from channels conversations.

"Our goal is to get a working IRC client that registers on the server and responds to pings."

The entire source code will be available at **https://github.com/Newky/irc_linuxvoice**. The code snippets here will not necessarily appear in the order they appear in the source code.

The goal is to get a working IRC client that registers on the server and responds to pings. First, we look at the high-level view from the main function, and then we'll go into deeper detail on each function.

```
def main():
  args = parse_args()
  sock = create_irc_socket(args.host, args.port)
  register(sock, args.nick, args.real_name)
  join_channel(sock, args.channel)
  read_loop(sock)
  sock.close()
```

First, the Python module **argparse** is used for parsing command line arguments. The defaults are configured to connect as we did with Telnet.

```
  parser = argparse.ArgumentParser()
  parser.add_argument("--host", dest="host",
    help="IRC host to connect to", default="irc.freenode.org")
  parser.add_argument("--port", "-p", dest="port",
    help="IRC port to connect to", default=6667)
  parser.add_argument("--nick", "-n", dest="nick",
    help="IRC nick to use.", default="LinuxVoiceTest")
  parser.add_argument("--real-name", "-r", dest="real_name",
    help="IRC real name to use.", default="Linux Voice IRC
Test")
  parser.add_argument("--channel", "-c", dest="channel",
    help="Channel for bot to join.", default="#linuxvoice")
  args = parser.parse_args()
```

Once the location of the server is established, the program establishes a connection with the IRC server. This is done in the **create_irc_socket** function.

```
def create_irc_socket(host, port):
  sock = socket.socket()
  sock.connect((host, port))
  return sock
```

This creates a socket, by default using the IPv4 address family as the default argument, which then connects to the given host and port. This is the raw socket that will be used for all interaction after this point. Once the connection is established, the program registers with the IRC server using the same registration flow as was used with Telnet. The **register** function is used to send both a **NICK** and a **USER** message. The actual message strings are handled by separate functions **nick_msg** and **user_msg**.

```
def nick_msg(nick):
  return "NICK {}\r\n".format(nick)

def user_msg(nick, real_name):
  return "USER {} 0 * :{}\r\n".format(nick, real_name)

def register(sock, nick, real_name):
  sock.send(nick_msg(nick))
  sock.send(user_msg(nick, real_name))
```

Before the program enters the **read_loop**, it joins a channel specified in the command line arguments. This is achieved in the **join_channel** function. This sends the **JOIN** message, which is constructed in the **join_msg** function.

```
def join_channel(sock, channel):
  sock.send(join_msg(channel))

def join_msg(channel):
  return "JOIN {}\r\n".format(channel)
```

A response is not read from the server at this point, as this is handled next in the **read_loop** function. The read loop is an infinite loop, which will break if a keyboard interrupt (Control + C) is detected. Each loop will read bytes from the socket, up to a maximum of 1024 bytes. However, if **\r\n** are not the last characters received (meaning that the message is not complete) it will continue to read until it has got a **\r\n** terminated string.

```
def read_loop(sock):
  try:
    while 1:
      data = sock.recv(1024)
      # if the data ends with a \r\n everything is fine.
      # if it doesn't we need to keep appending until it does
      while not data.endswith("\r\n"):
        data += sock.recv(1024)

      for message in data.split("\r\n"):
        # skip final empty string after split.
        if not message:
          continue
        source, command, rest = parse_message(part)
        action_on_commands(sock, source, command, rest)
  except KeyboardInterrupt:
    pass
```

The read buffer (in this case called data) is split into IRC messages using **\r\n** as the delimiter. Each message is then passed to the **parse_message** function, which splits it into source (if present), command and rest.

```
def parse_message(msg):
  # get rid of newlines and whitespace.
```

```
msg = msg.rstrip()
# split by space
components = msg.split()

# if the first part starts with a : the message includes a
# source.
if components[0].startswith(":"):
    source = components[0]
    command = components[1]
    rest = ' '.join(components[2:])
else:
    source = None
    command = components[0]
    rest = ' '.join(components[1:])

return source, command, rest
```

The purpose of an IRC bot is to carry out some action based on certain messages. When the program receives a message from the server, it has to decide what, if any, action must be taken. The **action_on_commands** function handles this; it takes the socket and the three arguments of the **parse_message** function as parameters and triggers an action based on how our program is set up.

```
COMMANDS = {
  "PING": pong_msg_responder,
}

def action_on_commands(sock, source, command, rest):
  responder_func = COMMANDS.get(command)
  # command has no responder defined:
  if not responder_func:
    return

  return responder_func(sock, source, rest)
```

Mission accomplished!

The function uses the **COMMANDS** dictionary to react to certain commands. The dictionary has commands as keys (such as **PING**) and functions as the values. This is one of the values of having functions as first class citizens in Python. All the functions should take **source** and **rest** as arguments. The **action_on_commands** function will look up the **COMMANDS** dictionary for the command; if it doesn't exist (for example if there is no **LEAVE** command configured), no action is taken and the function returns. However, if it does exist, it passes the socket along with the **source** and **rest** into the function.

Above, the **PING** command has been configured with the **pong_msg_responder** as its action function. This is a simple function which sends a pong message to the server with the content of the ping message as its parameters.

```
def pong_msg(body):
  return "PONG {}\r\n".format(body)

def pong_msg_responder(sock, source, rest):
  sock.send(pong_msg(rest))
```

With that, we have a working IRC client in that it will stay on the network until the program exits via a keyboard interrupt. But with this we can write some nice action functions to react to messages. Below, we add a way to greet people who join a channel. The **COMMANDS** dictionary is modified to add an action for the **JOIN** command, and the corresponding action function **join_msg_responder**.

```
COMMANDS = {
  "PING": pong_msg_responder,
  "JOIN": join_responder
}

def join_responder(sock, source, channel):
  sock.send(
    priv_msg(
      channel,
      'Welcome to {}, {}!'.format(
        channel,
        source
      )
    )
  )

def priv_msg(target, message):
  return "PRIVMSG {} :{}\r\n".format(target, message)
```

This is very similar to the **PING** example, but instead a **PRIVMSG** message is sent to the channel that the **source** has joined. To better illustrate this, an example of the original message would look like:

```
:Newky1~Newky1@127.0.0.1 JOIN #linuxvoice
```

Using this information, the program can craft a **PRIVMSG** message to send back to the channel specified in the parameters of the **JOIN** command. This is a very basic IRC bot. With some more commands added to the **COMMANDS** dictionary, it's possible to build a system which responds to keywords mentioned in users messages to channels. I look forward to seeing a range of IRC bots popping up on #linuxvoice which do all sorts of things.

RFC

An RFC is a "Request for Comments" document, which is authored by engineers or individuals involved with a specific technology. The IETF (Internet Engineering Task Force) adopts some of these documents as internet standards. They are in-depth documents which can help when writing an implementation of a standard such as IRC.

IRC has its own RFC as it is an open protocol. I encourage you to check it out if this article has piqued your interest in IRC. It is a document which has information on the general workings of an IRC server and client, while also going into finer detail on each of the supported commands. Given the scope of this article, we only cover the client portion of IRC, and only concentrate on some of the most common commands.

The basic IRC RFC can be found at **http://tools.ietf.org/html/rfc1459**. Additionally, since new versions of the IRC protocol have been released, there have been new RFCs released, but adoption of these are not widespread among IRC servers and clients. For your own interest, you can check out RFC 2810, RFC 2811, RFC 2812 and RFC 2813, which detail these new changes.

Richy Delaney is a software engineer with Demonware Ireland, working on back-end web services using Python and Linux. He has been an avid Linux user for the past five years.

LINUXVOICE TUTORIAL

KEEP AN EYE ON SHARES WITH SHARED CODE – PART 1

ANDREW CONWAY

Take big data from the stockmarket and parse it into something human beings can understand.

WHY DO THIS?

- Make sense of the sea of financial data.
- Do better with your savings than the paltry 1.5% offered by high street banks.
- Fund an extensive collection of red braces.

Many tools for processing data are based on free and open software (FOSS), especially if it's available online. That's because the people who write the software are often interested in such things. A few numerate techno-geeks even end up working for banks in the city analysing all kinds of stock market data. They are paid handsomely because their bosses make a tidy profit from financial transactions involving shares. In fact, the image of brace-wearing stockbrokers shouting and waving slips of paper on the exchange floor has become a thing of the past, having been largely superseded by clever algorithms that sprawl across the internet.

Before going any further, we'd like to be clear on one point: this tutorial is not on how to get rich quick by coding and hacking the stock market. Instead, we're going to focus on getting started with stock market data and examining the history of share price time series. Given the quantity of data involved we'll need to automate the process, and command line scripting is the ideal tool for the job. Next issue we'll move on to constructing some basic algorithms and writing code to evaluate trading strategies on past data.

The next step of trading on the real stock market requires either that you do some serious homework, or, better still, take professional advice on the risks and responsibilities involved. There's also the ethical consideration of whether speculating – frankly, gambling – on shares is harmful to the economy, causing bubbles that burst harming productive businesses. We'll deftly dodge such thorny issues and concentrate on working with the data.

Take the long view

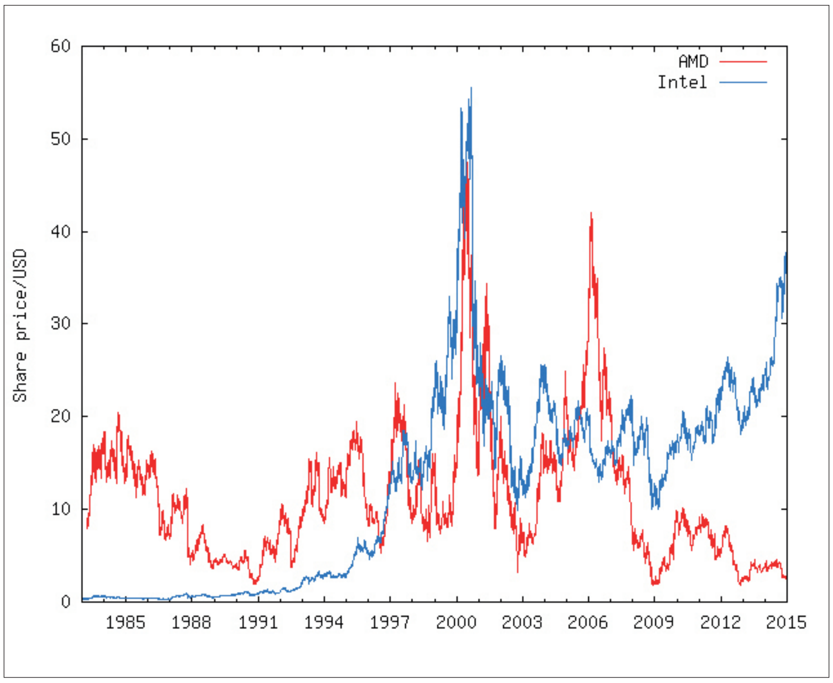
Before dipping your toe into the sea of data on the web, let's start by looking at the history of the FTSE 100, which is an index tracking the share price of 100 leading UK companies. It is calculated by taking the average over the total value of shares available for each of the 100 companies, and scaled so that it had a value of exactly 1000 when it began in 1984.

A convenient place to get the data is Yahoo's finance pages. Fire up your web browser and go to **https://uk.finance.yahoo.com/q/hp?s=^FTSE** and you'll see its present value, which will be updated live if trading is open on the London Stock Exchange. Select Monthly from the options on the right, and leave the date range on the left set at its default, which will get the entire history of FTSE 100 data, and click on the Get Prices button. The table will now be updated to show recent monthly values, but let's get the spreadsheet: scroll down to the bottom of the table and click Download To Spreadsheet.

You will now have a CSV (Comma Separated Variable) file called **table.csv** in your Downloads folder, but it's a good idea to rename it to something more meaningful, eg **ftse100.csv**. The structure is simple enough that you can open it up for a quick look in a text editor, such as *Geany* or *Kate*, or a pager on the command line, such as **less**. Clicking on the icon in your file manager will open it up in your default spreadsheet application.

You'll see that there are many rows, each one having a date, which will be the first working day of the month (stock exchanges are closed on weekends and public holidays). It will list the price when the stock exchange opened for trading, and when it closed, along with the highest and lowest values between times. For historical analysis, it's recommended to use the final column, which is called "Adj Close" – this is the close price adjusted to account for important information that came to light after trading.

We'll start by graphing the data using the standard spreadsheet approach because, for one thing, it's probably familiar to you, but also it highlights how inefficient this workflow is compared with



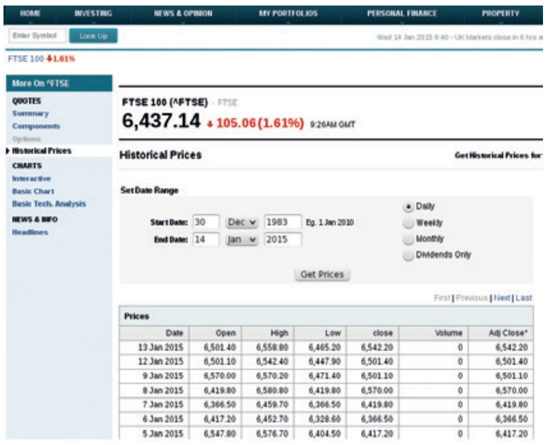
The share prices of AMD and Intel. Buying shares in either one in the early 1990s would have made you a great return if you sold at the 2000 peak, and although both fell sharply after the dot com bubble burst, and again in 2008, the Intel shares have been on an upward trend in recent years.

Stock market jargon

- **Shares** The ownership of a company can be split into many small pieces called shares. For example, if MyCorps Inc has a million shares and you have 500,000 of them, then you own half of MyCorps Inc. If the company makes a profit then it will pay out a share of the profit to you, called dividends, in proportion to your share ownership. So if the profit is $10m, you'll get $5m of it. But this isn't the only way you can make money. If a company is profitable, or people think it will be so in the future, then they will pay more for a share. So shares in MyCorps that you bought for $1 might become worth $2 and you can sell them and double your money.
- **The stock market** The stock market is a catch-all term for all trading in shares. Some companies only trade in private, but certain companies, called public companies, have shares that can be bought and sold at stock exchanges, such as those in London and New York. We'll concentrate on public companies because their data is released openly by the stock exchanges according to strict rules.
- **Stock market index** In order to judge how the stock market as a whole is behaving, indices are constructed that track the share prices of groups of large public companies. An increasing index means that most companies have share prices that are rising, and likewise a falling index means share prices are dropping. The most famous indices include the Dow Jones (USA), FTSE (UK), Nikkei (Japan) and NASDAQ (USA technology).

the command-line equivalent that we'll describe next. We're using a recent version (4.x) version of LibreOffice, but the procedure is common to many spreadsheet applications. Click on the A at the top of the Date column, then hold the Ctrl key down and click on G at the top of the Adj Close column. These columns should now be highlighted, but not columns B–F. Now click on the Chart button on the toolbar (or the Insert > Object > Chart menu item). For us, *LibreOffice* chose sensible defaults for this data and the graph it produced is shown in the boxout.

After inspecting the FTSE 100 graph, and especially after accounting for inflation, you might conclude that investing in shares has been a mug's game since the year 2000. But you might know that the wealthiest people have seen their wealth increase since 2000, and that they have their wealth invested in shares. This is thoroughly documented in Thomas Piketty's famous book *Capital in the Twenty-First Century*. How can we explain this apparent contradiction? The answer is in two parts. Firstly, remember that shares pay dividends, ie they divide the profits to shareholders, so there is a return even if the share price does not rise. Secondly, basing your portfolio on the FTSE 100 companies is a poor strategy; instead it's better to buy and sell shares from a wider pool to maximise returns. If you have the know-how to do this, or the wealth to pay a fund manager to do it for you, then it's possible that your portfolio will grow in value even when stock market indices are falling.



FTSE 100 (^FTSE) - FTSE

6,437.14 ↓ 105.06 (1.61%) 9:26AM GMT

Historical Prices

Get Historical Prices for

Set Date Range

Start Date: 30 Dec 1983 Eg. 1 Jan 2010

End Date: 14 Jan 2015

(• Daily) Weekly Monthly Dividends Only

Get Prices

First | Previous | Next | Last

Prices

Date	Open	High	Low	close	Volume	Adj Close*
13 Jan 2015	6,501.40	6,558.80	6,465.20	6,542.20	0	6,542.20
12 Jan 2015	6,501.10	6,542.40	6,447.90	6,501.40	0	6,501.40
9 Jan 2015	6,570.00	6,570.20	6,471.40	6,501.10	0	6,501.10
8 Jan 2015	6,419.80	6,580.80	6,419.80	6,570.00	0	6,570.00
7 Jan 2015	6,366.50	6,459.70	6,366.50	6,419.80	0	6,419.80
6 Jan 2015	6,417.20	6,452.70	6,328.60	6,366.50	0	6,366.50
5 Jan 2015	6,547.80	6,576.70	6,404.50	6,417.20	0	6,417.20

There are a few free finance services out there; Yahoo is just one, but it's particularly easy to extract data from.

Building a portfolio

Extending a portfolio beyond the FTSE 100 is clearly desirable. For example, if we chose to follow the FTSE 250, the index tracking the next 250 most significant UK-based companies, then we'd have enjoyed seeing its index rise from about 1,500 in 1986 to 16,000 in late 2014. A portfolio minimises risks by keeping your eggs spread across many baskets, but also gives you the freedom to swap shares in and out to maximise your returns. Quite apart from any profit incentive, there is the enticing challenge of constructing algorithms to manage the buying and selling of shares. But, before we can do that, we need to mine the huge history of time series data for information.

In case you're wondering if you've picked up a copy of *Stock Market Voice* by mistake, let's get back to Linux – specifically, the command line. Clicking around on stock market web pages can soon give you a headache from information overload, and that's assuming you can see the content – it's often delivered via Flash, Java and Linux-unfriendly plugins such as Silverlight. You can, as we did above, download CSV files from Yahoo and open them up as spreadsheets. But doing this for the thousands of shares available, and for many rows of data for each one, will gobble time and cause repetitive strain injuries to your mouse hand, eye and brain. Instead, the command line offers a viable path to automation of this workflow.

Let's start by looking at a famous company – ARM Holdings, which provides the CPUs for most phones and also the Raspberry Pi. To get its data from Yahoo, issue this command:

```
wget "http://ichart.finance.yahoo.com/table.csv?s=ARM.L" -O
ARM.L.csv
```

This sends a standard HTTP request asking for data on the share with abbreviation ARM.L and writes the returned data into the file **ARM.L.csv**. There will be a row in the CSV file for every day the share has been traded. As such, it's rather large, but you can use any of the following commands, and their variants, to help browse through the data:

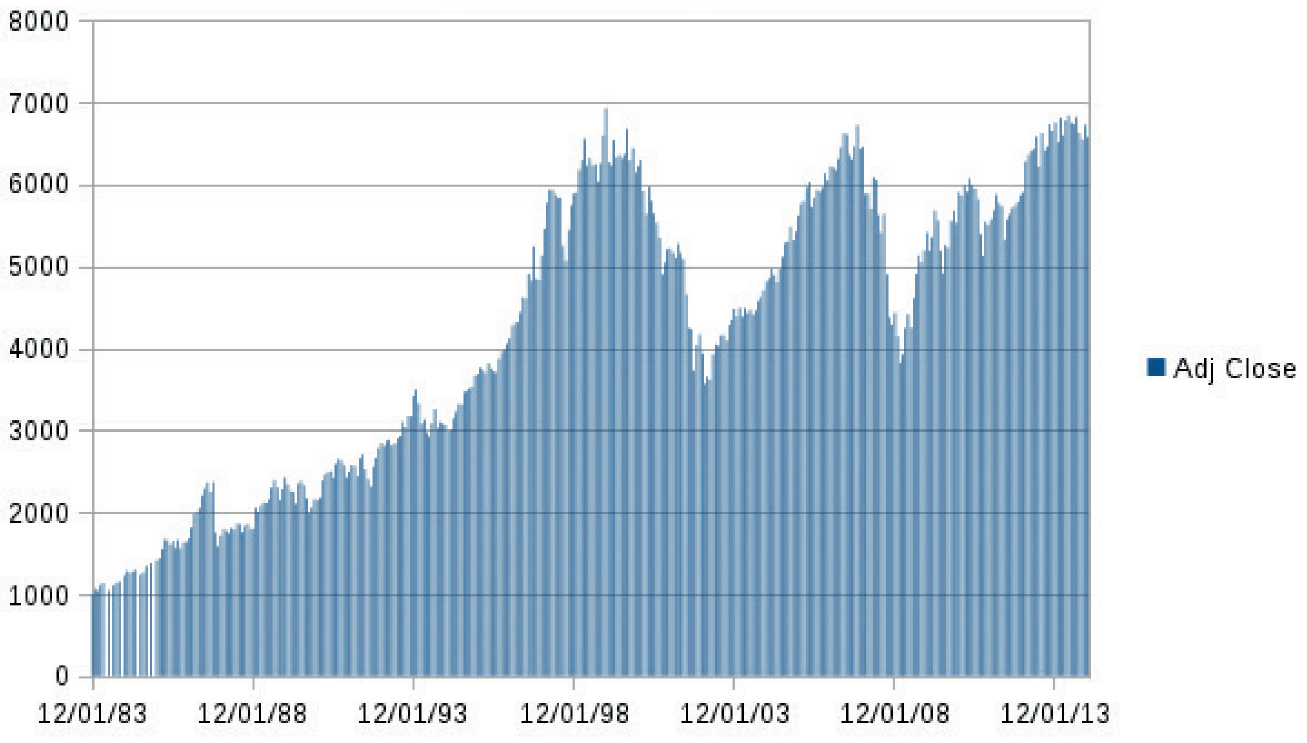
LV PRO TIP

Yahoo finance allows you to access its data service for free, but bear in mind that being free it may change or be withdrawn with little or no notice. Paid-for options are available if you need defined terms of service.

FTSE 100 history and inflation

This plot (generated by *LibreOffice*) shows that shares generally rose in price until the dot com crash in 2000, after which share prices recovered until crashing again in the recession of 2008. At the time of writing, the FTSE is back up to about 6500, close to its all-time high.

If you spent a sum of money buying a portfolio of FTSE 100 shares in early 1984, when the FTSE was at 1000, you would have received about six times that amount back if you sold them in early 2000, when it was about 6500. But if you'd bought them at the peak in 2000, then you would, at best, only break even if you sold them at any time thereafter. However, if you account for inflation – the effect of prices of everyday goods rising – a value of 6500 today is not the same as 6500 in 2000. Correcting for inflation, the FTSE 100 would need to be 10,000 if it were to equal its year 2000 peak in real terms.



The history of the FTSE 100 stock market index from when it began on 1 Jan 1984 to late 2014.

```
less ARM.L.csv
head -5 ARM.L.csv
tail -5 ARM.L.csv
cut -d, -f1,7 ARM.L.csv | less
```

The **less** command lets you page through the text of the CSV file. The **head -5** and **tail -5** lines show you the first 5 and last 5 lines of the file. The **cut** command splits each line using a comma delimiter and outputs only columns 1 (Date) and column 7 (Adj. close), which is then sent to **less** so you can page through it, though you could send it to **head** or **tail**.

If you want to download data for another company, you need only replace **ARM.L** in the above **wget** command with its stock market abbreviation. For example, Intel is INTC, and Apple is AAPL. The Yahoo Finance web interface has a Look Up field at its top left which will help you with this.

Let's take a first step in automating the data download. First, create a new directory and a list of shares that you wish to download and save it into **shares.txt**. You can list as many shares as you wish, but here's a concise example of just a few shares:

```
ARM.L
INTC
AMD
BRCM
```

Next, create a file called **download_shares.sh** containing these lines:

```
#!/bin/bash
URL=http://ichart.finance.yahoo.com/table.csv?s=
for share in $(cat shares.txt)
do
 wget "$URL"$share -O $share.csv
 sleep 2
done
```

To run this *Bash* script you'll need to make it executable:

```
chmod u+x download_shares.sh
```

so you can run it with

```
./download_shares.sh
```

The script takes each line of output from **cat shares.txt**, so **$share** becomes the abbreviation used in the **wget** line, which is used both to append to **$URL** and also to name the CSV file for output. The **sleep 2** line means that the script waits at least two seconds between sending requests – it's a free service so a bit of courtesy won't go amiss.

Once the data is downloaded we can use standard text processing commands to make short work of extracting interesting information. To see all share prices for a particular date:

```
grep 2014-12-01 *.csv | cut -d, -f1,7
```

Or to compare the share prices between two dates:

```
grep "2014-12-01\|2010-12-01" *.csv | cut -d, -f1,7
```

The output from this last command suggests that investing in ARM or Intel was a better bet than either AMD or Broadcom in recent years.

Commanding graphs

We can draw some pretty cool graphs from the command line with the venerable *Gnuplot*. If you don't have it, you can get it via your package manager for Debian-based distros with **sudo apt-get install gnuplot** or you can use **yum install gnuplot** for RedHat derivatives.

You can run *Gnuplot* in interactive mode, but as our goal is automation, we'll get straight to writing a quick script for it:

```
set datafile separator ","
set xdata time
set timefmt '%Y-%m-%d'
set xtics format "%Y"
set key off
plot filename every::1 using 1:7
pause -1 filename." Hit any ENTER to continue"
```

Save the file as **gnuplotter.gp**, then run the following command, which tells it to plot the data for Broadcom (though you can use any of the CSV files we've mentioned above) and you should see a window open with a plot of the data:

```
gnuplot -e "filename='BRCM.csv'" gnuplotter.gp
```

Note the single quotes inside the double quotes around the filename.

In the first line of the script, we tell it to use commas to separate values on each line of data. The next two lines say that the horizontal axis will be used for time data and the date format is specified. The **xtics** line tells it to display just the year (**%Y**) for labels, and the next line tells it not to display a key. Next we have the **plot** command, which instructs *Gnuplot* to graph the data with column 1 on the horizontal axis and column 7 on the vertical axis, and **every::1** makes it to skip the first line of the file, which is a non-numerical header. Finally, the **pause** command tells *Gnuplot* to wait until Enter is pressed before quitting – if you forget this line, the window with the graph will open then immediately close.

If you are a command line die-hard and are disappointed that you have to leave the terminal window, put this line at the start of **gnuplotter.gp**:

```
set term dumb
```

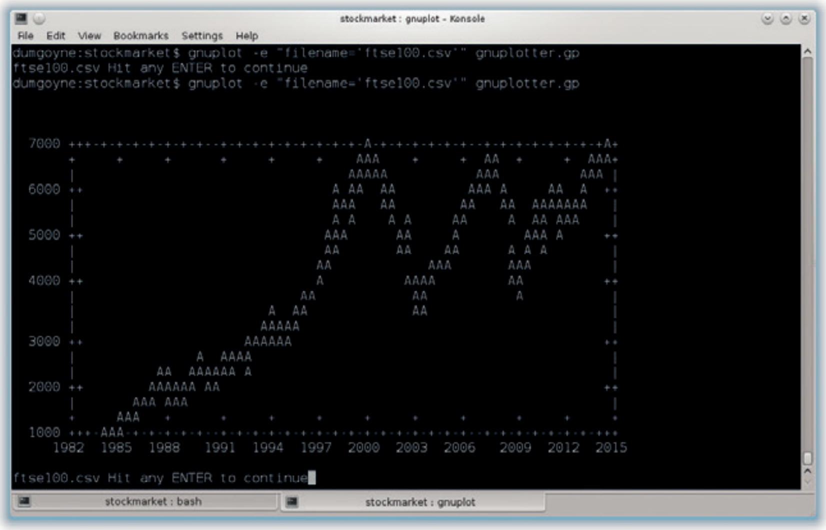
and a graph will be plotted with text characters.

Bring it all together

Let's put what we've seen above into one script so we can quickly review graphs of shares in your portfolio. Enter the following lines in a file called **review_shares.sh** and save it in the same directory as **gnuplotter.gp** and your downloaded CSV files:

```
#!/bin/bash
for share in $(cat shares.txt)
do
 head -5 $share.csv
 gnuplot -e "filename='$share.csv'" gnuplotter.gp
done
```

Make it executable with **chmod** and run it, just like we did above for **download_shares.sh**. Each plot will be displayed in turn and you just hit the Enter key to show the next plot. The **head -5** line shows the most recent data to accompany the graph, but you can add lines to display whatever information you wish to see on your portfolio.


stockmarket : gnuplot - Konsole
File Edit View Bookmarks Settings Help
stockmarket : bash
stockmarket : gnuplot

```
dumgoyne:stockmarket$ gnuplot -e "filename='ftse100.csv'" gnuplotter.gp
ftse100.csv Hit any ENTER to continue
dumgoyne:stockmarket$ gnuplot -e "filename='ftse100.csv'" gnuplotter.gp



  7000 +++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-A-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+A+
       +      +     +      +     +     +    AAA    +     +  AA  +     +  AAA+
       |                                   AAAAA          AAA           AAA |
  6000 ++                                A AA  AA        AAA A     AA  A  ++
       |                                 AAA   AA       AA   AA  AAAAAAA    |
       |                                 A A    A A    AA     A  AA AAA     |
  5000 ++                               AAA      AA    A       AAA A       ++
       |                                AA       AA   AA      A A A         |
       |                               AA          AAA        AAA           |
  4000 ++                              A         AAAA         AA           ++
       |                             AA           AA          A             |
       |                         A  AA            AA                        |
       |                        AAAAA                                       |
  3000 ++                     AAAAAA                                       ++
       |                 A  AAAA                                            |
       |            AA  AAAAAA A                                            |
  2000 ++          AAAAAA AA                                               ++
       |         AAA AAA                                                    |
       +       AAA    +      +     +     +     +     +     +     +     +    +
  1000 +++-AAA-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+++
          1982  1985  1988   1991  1994  1997  2000  2003  2006   2009  2012  2015

ftse100.csv Hit any ENTER to continue
```

A graph of the FTSE 100 data rendered by *Gnuplot* in a terminal window. Much detail is lost compared to its graphical equivalent (see above), but it clearly shows the long-term trends of this time series.

Next steps

Although the command line tools such as **grep** and **cut** are powerful, they deal only with text and can't perform numerical operations such as finding minimum and maximum values or calculating averages. If you want to stay close to the command line and augment the above scripts to provide statistics to accompany the graphs, then you could try using **awk** – it can perform the text functions of **grep** and **cut** but can also perform numerical operations. Beyond that, using a full-blown programming language such as Python, Ruby or Perl is probably best.

There is a huge amount of information to digest on the Yahoo Finance pages, and we've only scratched the surface of what's on offer. It's well worth spending time browsing through it, and if you're database minded you might find its YQL facility interesting, and if you're into web apps, there's an API to play with. It's also worth going to the horses' mouths and visiting the websites of various stock exchanges. If you'd like to experiment with simulated buying and selling of shares using real data, there are mobile apps such Stock Trainer that will give you a feel for the workings of the stock market without risking any of your money.

Next issue we'll turn to turn our attention to devising algorithms to decide when to buy and sell shares.

Andrew Conway, millionaire philanthropist, tracks the stars to predict the future – just like real economists!

LINUX VOICE TUTORIAL

A BRIEF HISTORY OF COMPUTER LANGUAGES

JULIET KEMP

Or: a whistle-stop tour through ways people have talked to computers, before we kick off with more in-depth tutorials.

Ada Lovelace wrote the first ever computer program without even having a computer. She wrote out a detailed description of how one would calculate Bernoulli numbers on Babbage's Analytical Engine (sadly never built). She, and others, also considered punch cards as a means of encoding the instructions to an Engine-like device. For more on this, see LV001 (if you haven't seen it yet, you can download the full PDF from here: **www.linuxvoice.com/download-linux-voice-issue-1-with-audio**).

The first computers that were successfully built, in the 1940s, were programmed in machine code, or at best in assembly language, with mnemonics rather than numeric codes. Whilst this lent itself to a high degree of fine-tuning, it was also error-prone and very hard work.

Those early programmers were interested in developing high-level languages; languages that were abstracted from the details of the machine doing the work. This allows a programmer to specify what they want without worrying about the details of memory location, chip instructions, and so on. The first high-level language ever designed was Zuse's Plankalkül, but as he lacked a functional computer at the time it wasn't implemented until over 50 years later.

The early 1950s saw several more attempts. Short Code (designed by John Mauchley) was implemented for UNIVAC, and aimed to make mathematical expressions easier to code. However, it was interpreted rather than compiled, so had to be translated every time, running about 50 times slower than assembly. The first compiled language was Autocode, developed at the University of Manchester for the Mark 1. Grace Hopper's FLOW-MATIC was a couple of years later, and was aimed at business customers who might be uncomfortable with mathematical notation. None of these are still in use today; but their successors, Fortran, Lisp and COBOL, have all survived.

Fortran was created in 1957. It had 32 statements, and was of course stored on punchcards, one card per line of code. Compilers for many different computers were rapidly developed due to its growing popularity, making it arguably the first cross-platform language. Here's an example (save as **hello.f95**):

```
! Hello World
program hello
 print *,"Hello World"
end program hello
```

Install the gfortran package, and compile it with **f95 -o hello hello.f95**, then run it with **./hello**.

Lisp emerged in 1958. The name derives from LISt Processing, and Lisp is heavily list-based (and, famously, involves a lot of brackets). Here's a Hello World (save as **hello.lisp**):

```
(write-line "Hello World")
```

Install **sbcl** (if you want to do much Lisp development you'll also want to install and set up Slime (the Superior Lisp Interaction Mode for Emacs) and *Emacs*), and run this with **sbcl --script hello.lisp**.

COBOL was designed in 1959–1960, by a steering committee, and took a lot of features from FLOW-MATIC. It was intended to be verbose and easy to understand for non-experts, and to be highly flexible for multiple uses. Although it's often derided, a significant number of large organisations still have COBOL legacy code on mainframes. Here's a Hello World example (save as **hello-cobol**):

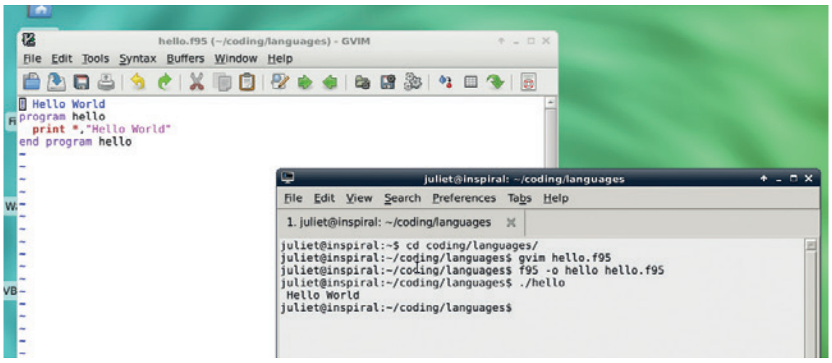
```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-COBOL.
PROCEDURE DIVISION.
 DISPLAY 'Hello World!'.
 STOP RUN.
```

Install the **open-cobol** package, compile with **cobc -free -x -o hellocobol-exe hello-cobol** and execute.

After these first language pioneers, as the 1960s and 1970s progressed, more languages, and more programming theory, began to develop; along with various distinctions and coding structures (some of which overlap with one another).

Array programming

The basic idea of array programming is to apply an operation to a range of values at the same time. So an operation will, instead of adding two single numbers, add two arrays (or vectors, or matrices, or other


hello.f95 (~/coding/languages) - GVIM
File Edit Tools Syntax Buffers Window Help
juliet@inspiral: ~/coding/languages
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/languages

```
! Hello World
program hello
  print *,"Hello World"
end program hello
```

```
juliet@inspiral:~$ cd coding/languages/
juliet@inspiral:~/coding/languages$ gvim hello.f95
juliet@inspiral:~/coding/languages$ f95 -o hello hello.f95
juliet@inspiral:~/coding/languages$ ./hello
 Hello World
juliet@inspiral:~/coding/languages$
```

Fortran code and code compiling/running, again. The ! line is a comment.

grouped data, depending on the language and the problem being handled). This is particularly useful for mathematicians, who often want to deal with grouped data like this.

Given that early computing was closely linked with mathematics, it's not surprising that dealing with arrays was of immediate interest. Fortran had some array handling from the start, but was more multi-purpose, and full array handling wasn't introduced until Fortran 90. APL (A Programming Language), one of the best-known array programming languages, was developed between 1957 and 1967. It was explicitly intended to provide a language for applied mathematics. It uses a multi-dimensional array as its basic data type, and has special characters to represent specific operations. This makes for code which is concise, but baffling to read for the non-expert. Here's an example:

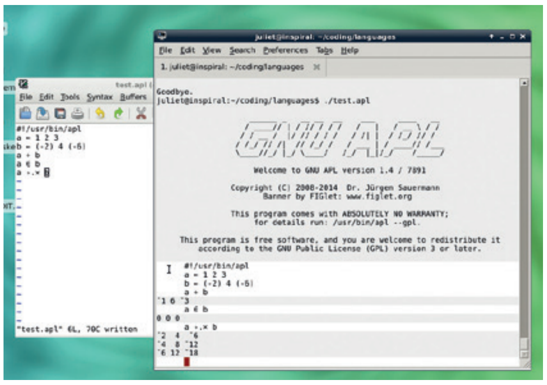
```
#!/usr/bin/apl
a ← 1 2 3
b ← (-2) 4 (-6)
a + b
a ∈ b
a ° . × b
```

APL uses ← (Unicode leftward arrow) for assignment. a + b simply adds.The **∈** character (Unicode element of) returns 1 if the element in position ***n*** is the same in both a and b, and 0 otherwise. Finally, - . is the outer product operator, which applies a specific operation to all the combinations of the elements of the operands. So here, we multiply (× – not the character x but the Unicode multiplication character) **b** by each of the elements of **a** in turn.

Run with **./test.apl**, to get the output

```
-1 6 -3

0 0 0

-2 4 -6
-4 8 -12
-6 12 -18
```

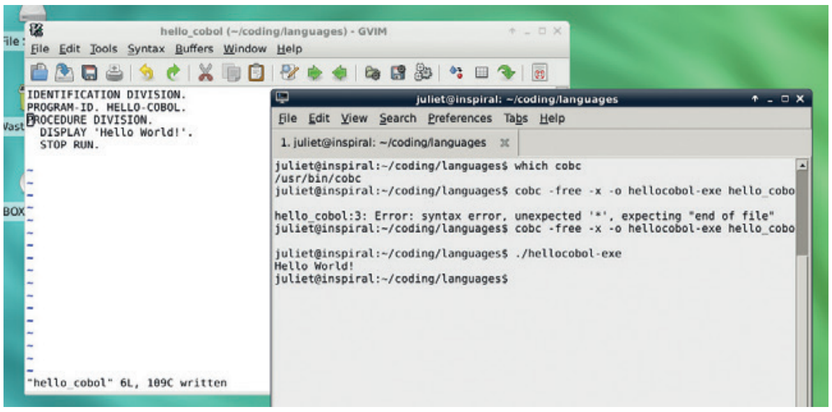
The supercomputers of the 1960s and 1970s were designed to handle vectors and arrays with particular



```
#!/usr/bin/apl
a ← 1 2 3
b ← (-2) 4 (-6)
a + b
a ∈ b
a ∘.× b
"test.apl" 6L, 70C written
```

```
Goodbye.
juliet@inspiral:~/coding/languages$ ./test.apl

Welcome to GNU APL version 1.4 / 7891

Copyright (C) 2008-2014  Dr. Jürgen Sauermann
Banner by FIGlet: www.figlet.org

This program comes with ABSOLUTELY NO WARRANTY;
for details run: /usr/bin/apl --gpl.

This program is free software, and you are welcome to redistribute it
according to the GNU Public License (GPL) version 3 or later.

      #!/usr/bin/apl
      a ← 1 2 3
      b ← (-2) 4 (-6)
      a + b
¯1 6 ¯3
      a ∈ b
0 0 0
      a ∘.× b
¯2  4  ¯6
¯4  8 ¯12
¯6 12 ¯18
```

Both *Vim* (using digraphs) and *Emacs* (using an appropriate Lisp file and chords) will support APL characters.



```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-COBOL.
PROCEDURE DIVISION.
  DISPLAY 'Hello World!'.
  STOP RUN.
"hello_cobol" 6L, 109C written
```

```
juliet@inspiral:~/coding/languages$ which cobc
/usr/bin/cobc
juliet@inspiral:~/coding/languages$ cobc -free -x -o hellocobol-exe hello_cobo

hello_cobol:3: Error: syntax error, unexpected '*', expecting "end of file"
juliet@inspiral:~/coding/languages$ cobc -free -x -o hellocobol-exe hello_cobo

juliet@inspiral:~/coding/languages$ ./hellocobol-exe
Hello World!
juliet@inspiral:~/coding/languages$
```

Our simple COBOL example running in the right-hand window (note syntax error on first compile!).

ease. Other modern array programming languages include J, MATLAB, and S-Lang.

Imperative vs declarative programming

Imperative programming involves issuing a series of commands to the computer. At the hardware level, almost all computers operate in an imperative style, with machine code consisting of instructions operating on memory contents. Initially, computer programmers were using machine language, and thus an imperative style; so the first high-level languages (such as Fortran and COBOL) were similar. It does make a certain cognitive sense to write your code as a set of algorithmic steps.

Declarative programming takes a different approach. Instead of telling the computer how to perform a task (what steps to take), it describes what computation should be carried out, and the compiler then translates this into specific steps. Pure declarative programming also avoids "side effects" (functions that modify state rather than just returning something) and has immutable variables. Imperative programming, on the other hand, makes frequent use of side effects, and happily alters variables.

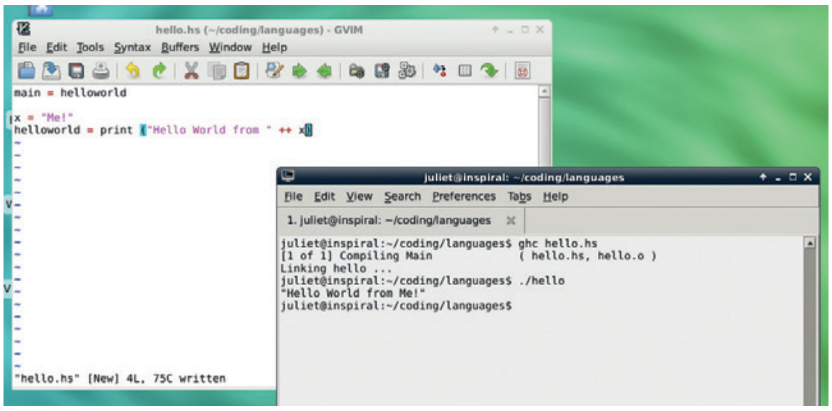
If you've ever written any SQL, that's declarative: you describe the result you want (a certain selection of records), and the code chooses how it returns that result. In general, many languages can be written in either an imperative or a declarative style, although some are much more inclined one way than the other. C, for example, is highly imperative; Haskell is highly declarative (as are other functional languages).

If your main experience is in imperative programming, the declarative approach can feel awkward. Here's a Python example of doing the same thing in two different styles. You can try it out in a browser at **www.skulpt.org**.

```
# Declarative
fours = [x for x in range(100) if x%4 == 0 ]
print fours

# Imperative
fours = []
for i in range(100):
  if i%4 == 0:
    fours.append(i)
```

LV PRO TIP

Reconfigurable computing is not imperative at machine-code level; there is a main processor which reconfigures subsidary systems on the fly to suit the requirements of the instructions provided.


hello.hs (~/coding/languages) - GVIM
File Edit Tools Syntax Buffers Window Help
juliet@inspiral: ~/coding/languages
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/languages

```
main = helloworld

x = "Me!"
helloworld = print ("Hello World from " ++ x)
"hello.hs" [New] 4L, 75C written
```

```
juliet@inspiral:~/coding/languages$ ghc hello.hs
[1 of 1] Compiling Main             ( hello.hs, hello.o )
Linking hello ...
juliet@inspiral:~/coding/languages$ ./hello
"Hello World from Me!"
juliet@inspiral:~/coding/languages$
```

Haskell is a functional, rather than a declarative language.

```
print fours
```

(With thanks to Mark Rushakoff.) The first one says what you want – numbers which are multiples of 4 (% is the modulo operator) – and the implementation is up to the machine. The second one describes exactly how to construct the required array.

Functional programming

Functional programming is declarative; functional languages construct and evaluate functions and treat data as unchangeable. So you can't alter data in-place; instead you have to apply a function to one data item and store the result in another data item. It's based on lambda calculus, which is a theoretical mathematical framework for describing functions. It will also use higher-order functions (functions that operate on functions) where an imperative language might use a loop.

The avoidance of side-effects makes programs easier to verify and to parallelise. However, some things (I/O being one notable example) do seem best suited to some kind of state approach. Functional languages will 'fake' this in various ways: Haskell uses monads; other languages use data structures to represent the current state of a thing.

Lisp was the earliest functional-type language; it was followed by APL (see above) and ML, which has various offshoots. Probably the best known modern functional language is Haskell. There's a Haskell example above; here's another one (save as **hello.hs**):

```
main = helloworld
x = "Me!"
helloworld = print ("Hello World from " ++ x)
```

Compile and run it with **ghc hello.hs**; **./hello. helloworld** is a function, and **x** is a variable, but they can both be defined in the same way. **main** is the main program control structure (what runs when you execute the compiled file). There's lots more information and tutorials on the Haskell web page.

Structured programming

Structured programming argues that programs are composed of three control structures:

1. **Sequence** A set of statements or subroutines ordered in a particular sequence.
2. **Selection** A statement or statements executed based on the program state (eg **if/then** structures).
3. **Iteration** A statement executed until a certain condition is achieved (eg **while**, **for**, **do/until**).

Blocks and subroutines group statements together. The structured program theorem states that these, when combined, are sufficient to describe any computable function.

Non-structured programming simply has a sequence of commands, although usually these are labelled so that execution can jump to that point. Loops, branches, and jumps exist, and sometimes basic subroutines.

Today, all high-level languages have some form of programming structure (including older languages like Fortran, COBOL, and BASIC), and structured coding is the norm. But early programmers were accustomed to machine code or assembly language, which had only an ordered sequence of commands. A single statement in a high-level language will be spread over multiple statements in assembler. Assembly language coders were skilled at manipulating code in complex and highly efficient ways and it seemed far from obvious that all of this was even conceivably structurable. COBOL was notoriously unstructured and made extensive use of GO TO statements. Edsger Dijkstra's letter Go To Statement Considered Harmful is probably best-known contribution to the debate.

Structured programming at its most basic means writing code that looks like this:

```
$a = 3;
```

LV PRO TIP

The **haskell-platform** package is available via the package repository of several distributions; check out the Haskell platform webpage for details for your distro or to build from source.

Interpreted vs compiled

In the very early days, programs were neither interpreted nor compiled. Instead, programmers wrote machine code, which ran directly on the hardware. Once languages began to be developed, the distinction between compiled and interpreted developed alongside them.

Broadly speaking, a compiled language is one in which the instructions written by the programmer are translated (by the compiler) into machine code all at one go. The compiled program can then be run on the machine. Parse everything, then run it.

Interpreted languages, in contrast, are read at runtime by another program, an interpreter, which then translates each instruction, one at a time, into machine code. So parsing and execution happen at the same time. Parse a statement, run it, parse the next statement, and so on.

In modern languages, the distinction can be quite blurry. Some modern compilers can parse and execute in memory, so although the steps are distinct, the programmer issues only one command. Other languages compile to virtual machine bytecode, which is another step (or more!) away from the metal. Ultimately, any program has to be translated into machine code; the question is how that process occurs and which steps occur in what order.

Interpreted and compiled languages both have their advantages and disadvantages; as ever, it's about using the best tool for the job.

```
$pi = 3.14;

sub area($_) { return $_[0] * $pi * $pi; }

print area($a);
```

instead of code that looks like this:

```
print 3 * 3.14^2;
```

The second might be shorter, but it is less reusable and less maintainable.

Procedural programming is derived from structured programming, and is based on the idea of procedures (or methods, or functions), consisting of a series of steps. It is often contrasted with object-oriented programming (OOP).

Object-oriented programming

It's nearly impossible these days not to have encountered OOP (whether or not you like it). Many modern languages are multi-paradigm and support OO alongside a more imperative style (eg Perl, PHP, Python). Java, in contrast, is exclusively OO.

The basic OOP idea is to fold both code and data into objects with behaviour (code) and state (data). Code is executed by creating an object and causing it to behave in a particular way. So to add X and Y, you would pass X into Y's "add" method (**Y.add(X)**).

Objects can inherit methods and data from one another, so OO languages have a class (object) hierarchy. OOP encourages modular programming (though non-OO languages can also support modules), intended to simplify code reuse, by bundling together objects and everything associated with them.

Some of the advantages claimed for OOP are:

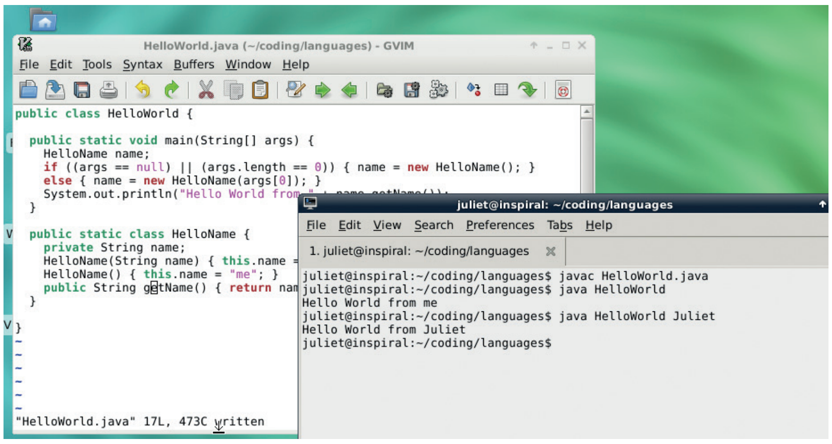
- Improved code reusability; great for modules and code libraries.
- Interfaces and encapsulation make it easier to use others' code; you need only understand the interface, not the details of the code.
- Encapsulation makes it easy to hide values that shouldn't be changed.
- Improved code organisation and simpler syntax.
- Forces better advance planning, and is easier to maintain afterwards.

There are also, of course, disadvantages:

- OO programs tend to be large. This is less of a problem on modern machines with lots of memory and hard drive space.
- Programs are often slower, although again with modern machine resources this is less important.
- More effort required up-front, which some may consider wasteful. The larger the project, the less true this is, but for a small project the effort may be overkill.
- Lots of code boilerplate. (This is less hassle with a decent IDE.)

Both advantages and disadvantages are true; which way they balance will depend on the project and the people working on it.

Here's a brief Java example (save as **HelloWorld.java**):


HelloWorld.java (~/coding/languages) - GVIM
File Edit Tools Syntax Buffers Window Help
juliet@inspiral: ~/coding/languages
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/languages

```
public class HelloWorld {

  public static void main(String[] args) {
    HelloName name;
    if ((args == null) || (args.length == 0)) { name = new HelloName(); }
    else { name = new HelloName(args[0]); }
    System.out.println("Hello World from
  }

  public static class HelloName {
    private String name;
    HelloName(String name) { this.name =
    HelloName() { this.name = "me"; }
    public String getName() { return nam
  }

}
~
~
~
~
~
~
"HelloWorld.java" 17L, 473C written
```

```
juliet@inspiral:~/coding/languages$ javac HelloWorld.java
juliet@inspiral:~/coding/languages$ java HelloWorld
Hello World from me
juliet@inspiral:~/coding/languages$ java HelloWorld Juliet
Hello World from Juliet
juliet@inspiral:~/coding/languages$
```

Our **HelloName** example in Java runs both with and without a name provided on the command line.

```
public class HelloWorld {
 public static void main(String[] args) {
  HelloName name;
  if ((args == null) || (args.length == 0)) { name = new
HelloName(); }
  else { name = new HelloName(args[0]); }
  System.out.println("Hello World from " + HelloName.
getName());
 }

 private static class HelloName {
  private String name;
  HelloName()          { this.name = "me"; }
  HelloName(String name) { this.name = name; }
  public String getName() { return name; }
 }
}
```

This is deliberately a little verbose to demonstrate objects. The **HelloName** object stores a name; you could easily extend it to store more data. The **HelloWorld** class creates the new object, then uses the **getName()** method to retrieve the string. Compile it with **javac HelloWorld.java** and run it with **java HelloWorld NAME**.

More out there...

There are a bunch of types and areas of programming language I haven't been able to cover here, including systems programming, logic programming, reflection programming; ideas around modularity, security, concurrency, and other aspects of modern computing also inform current thinking.

Language development continues at a fair old clip, to the point that any attempt to list languages available as I write would probably be out of date by the time we go to print. And all of them – all the languages, all the paradigms, all the tweaks and mechanisms and constructs – have their places where they're useful and their places where they don't fit. One of the joys of programming is just how many options there are out there to explore.

LV PRO TIP

Java is available from your distribution's package manager or from the Java website. Make sure you install the Java Development Kit (JDK) as well as the Runtime Environment (JRE).

Juliet Kemp is a scary polymath, and is the author of Apress's *Linux System Administration Recipes*.

LINUX VOICE TUTORIAL

ARDUINO HARDWARE ENABLEMENT (PART DEUX)

NICK VEITCH

In which we complete our project to connect a cheap OLED display into our Arduino, while building a driver and learning some C++.

WHY DO THIS?

- Learn how to plug awesome displays into cheap hardware and code your own interface using the Arduino IDE and a smattering of learner-level C++.

Last month, we covered the schematics of the project, using the I2C library as an interface and writing some driver code to make it easy to display things. Now we're going to play around with the display, starting with the long-promised splash screen. This is non-essential, but nice, as it means when the display starts up, you can see that it is actually working. It is also a useful way of breaking into the topic of displaying stuff on the screen.

The first thing is to have some data to put on the screen. We can convert an image into a bitmapped array of chars we can write to the screen (see Python To The Rescue box for how we generate this data). If you think that storing a huge array of chars rather flies in the face of our intention to save runtime memory then you are very right. That's why we aren't going to put it in runtime memory. We are going to store it in the program memory, by using the **PROGMEM** macro (see the Memories boxout for why this works).

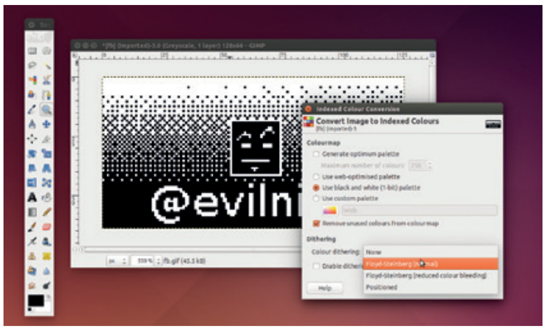
So, in our **.cpp** file, we simply need to define a giant array of bitmap data:

```
const char fb[1024] PROGMEM = {
B00000000, B00000000, B00000000, ...
...
}
```

Obviously, we are not going to show the full 1k of data here; this is just to show you how it is constructed (think yourself lucky we aren't still living in the age where you had to type in listings from magazines. This way, it is only my fingers which hurt).

All we need to do now is write a function that will read this data from the flash memory and send it a byte at a time to the display. This is made slightly more complicated by a) the page addressing mode of the device and b) the fact that we stored the data in flash memory.

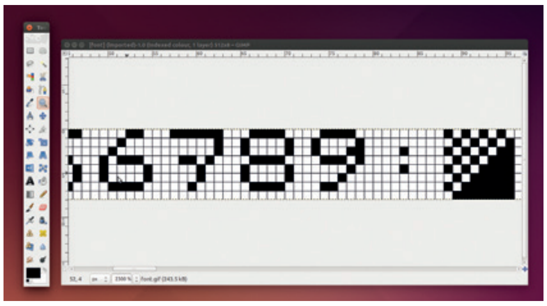
The page mode means that we need to read the data 8-bit lines at a time until we have filled an 8x8



Indexed Colour Conversion
Convert Image to Indexed Colours
Colourmap
- Generate optimum palette
- Maximum number of colours: 256
- Use web-optimised palette
- Use black and white (1-bit) palette [✓]
- Use custom palette
- Remove unused colours from colourmap [✓]

Dithering
- Colour dithering: None / Floyd-Steinberg (normal) / Floyd-Steinberg (reduced colour bleeding) / Positioned
- Enable dithering
- Help

It is a good idea to convert to a bitmap first before you run the converter, otherwise the dither patterns may surprise you.



When you make your own font, you can choose how to represent a 7. And other things. Just make sure you use consistent spacing.

square, then move on to the next. When we get to the end of a line (after 16 blocks) we have to set the page index to the next value. The command to address page0 is '**B0**', and thankfully, the other page address commands follow sequentially, so we can simply loop and address 'B0 + offset' (literally, if we make 'offset' a char value).

Each 'page' is then 16 x 8 bytes of data, or 128 bytes. We can make a loop that reads the next 128 bytes from our declared array and sends it to the display. If we keep an independent index value, we don't need to come up with complex maths to work out where in the data we should be. Like this:

```
for (int n=0;n<128;n++)
        {
        sendByte(pgm_read_byte(&(fb[index++])));
        }
```

This uses the special **pgm_read_byte** function to read from flash memory (check the Memories boxout for more details on this and why we use the **&( )** construct), and increments the **index** variable each time it is called so we can read the image data sequentially. The rest of this whole function is just another loop which goes through the pages of display memory, and the corresponding commands to the display to set the usual stuff (page mode, start and end of data, slave address and such):

```
void evilOLED::splash ()
{
        int index = 0;
        sendCmd(0x00);// -> page mode
        sendCmd(0x00);// lower page
        sendCmd(0x07);
        for (int p=0; p<8;p++){
        sendCmd(0xb0 + p);
```

Buffered side down

So, what are framebuffers, why would you want one, and why do I specifically not want one?

A framebuffer is a buffer, for a frame. In this context, the frame is a graphical one. The framebuffer is basically just an exact copy of the memory containing the image or text you want to display. Framebuffers are often very useful and desirable. You may want to buffer video frames in order to display them quickly (you don't have to draw anything, just dump all the memory to the display device) or for drawing effects which require knowing what is currently on the screen.

As memory tends to be written in bytes (and the device we're looking at here is no exception), if you want to write a single pixel somewhere, you need to know which other pixels in the same byte of memory are already turned on (otherwise you will be overwriting them). In many cases, you may find yourself communicating with a display device to ask it what is in its display memory so you can then superimpose your pixel and write the whole byte back. This is time consuming (in terms of actual processor time, not just because you have to write extra code) and sometimes not actually possible. The particular display we are using doesn't have a read mode for I2C communication – there is no way of reading the display memory.

In that case, you may surmise, having a framebuffer is a good idea. You can keep a copy of what should be on the display, make your adjustments to that and then just update the bits of the display memory that are required.

That is a reasonable suggestion, and one used by many device enablement libraries. However, there is a problem with this approach. A 128x64 display requires a framebuffer capable of holding 8192 bits of data, which is 1024 bytes of data. That doesn't sound like a lot, but depending on which Arduino you are using, you may have as little as 2560 bytes of dynamic memory available. That means the framebuffer will take up nearly half of your working memory. Sure, this isn't too much of a problem if all you are doing is displaying something on screen, but what if you're trying to do something else with your project – what if you need to address other hardware like GPS or network devices which can also consume a lot of RAM? in such circumstances, using half your RAM just to be able to display the output is a little annoying.

No buffers

What is the answer then? It turns out that it is to borrow an idea from the past, and simply writing to the screen in predefined graphical blocks or characters. If we define characters we want to use on screen, and imagine that we will never want to combine characters in the same space, then we can happily overwrite the contents of the display memory without needing a copy of it. Essentially, we take the 128x64 pixel display and turn it into a 16x8 character display (although, of course you could define characters of any size if you wanted, this is just assuming 8x8 characters). This does mean you lose the ability to plot individual pixels or create arbitrary shapes, but on the plus side it saves a lot of memory and results in faster display updates. There is also nothing stopping you from creating a font containing all sorts of useful symbols or graphical elements if you want them.

```
        sendCmd(0x00);
        sendCmd(0x10);
        dataStart();
        sendByte(0x78);
        sendByte(0x40);

        for (int n=0;n<128;n++)
        {
        sendByte(pgm_read_byte(&(fb[index++])));

        }
        dataStop();

        }
}
```

So, as well as making a nice splash screen, that's how you dump a whole screen of data to the display.

Font pun goes here

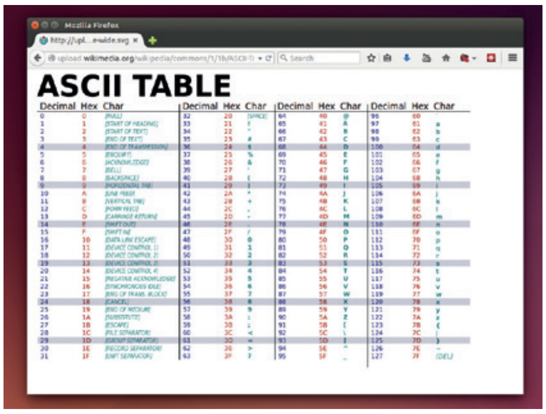
A screen of data is one thing, but we promised we'd cover character mapping the display. This is a similar but not the same procedure – in this case, you want to fill one of those 8x8 blocks at a time. Of course, you could choose a different size, but 8x8 is very convenient for the way the device memory is organised, and also a good size for a minimalist text font. We can construct the font as a bitmap, in more or less the same way as the splash screen, except this time we want to have it only 8 pixels high by however long we need.

However long we need may be a surprise, but it falls in line with our space-saving ethos. Sure, we could implement a full ASCII-type character set. But fully half of those characters are useless (**SYNCHRONOUS IDLE** anyone) and if we were to map them all, we'd be using up another 256x8 bytes, or, yes, another 1k of precious memory.

If you want to do this, then there are plenty of fonts you can crib from various libraries that already exist (check out **www.henningkarlsen.com/electronics/r_fonts.php** for some good examples, or fish through **https://github.com/adafruit/Adafruit-SSD1351-library** to find the Adafruit ones.

Defining your own font means you can make your own symbols. If you want a battery or network indicator, you can just draw one. Bar displays become just 8 characters worth of progressively filled blocks, and you can still have text and numbers. The example font included with the source for this tutorial has some strange symbols, but also the standard numbers and upper case letters, and all in half the amount of space a standard font would take up.

The only real trick here is that you want to make it easy to convert from 'normal' text in your Arduino code (which is ASCII) to your custom font. For the example font, this is achieved by ignoring the first 47 characters and starting with '0'. This means that we can more or less easily convert from ASCII to custom



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Google a standard ASCII table and you will see most of the characters are rubbish. And who needs lower case!

font just by shifting everything so many spaces back. Ah. Space. The space character is one we have left out, so we have to add a special capture routine to remap that. It isn't as tricky as it sounds:

```
void evilOLED::putChar(char c)
{
        //assumes cursor is set to correct place
        if ( c == 0x20) // trap SPACE character
        {
        c = 0x30; //this is where I put it
        }
        else
        {
        c -= 0x30; //remove offset to match ASCII value
        }
        dataStart();
        sendByte(0x78);
        sendByte(0x40);
        for (char i=0; i<8; i++)
        {
        sendByte(pgm_read_byte(&(font_bmap[8*c+i])));
        }
        dataStop();
}
```

Memories

Not all memory is created equal. For a start, there are all different sorts of memory – SRAM, EEPROM, DRAM, PSRAM and so on. There are enough different types to fill a few pages just defining them all. However, we are more interested in how memory is used, and how in particular, your Arduino handles memory.

Depending on the model of Arduino, you will have a certain amount of EEPROM memory, an amount of Flash program memory and some dynamic memory – which, confusingly enough, uses SRAM (static random access memory) on the AVR chip itself. The 'Static' here refers to the way the memory is built; it uses bistable latches so it doesn't need to be refreshed periodically like DRAM.

The standard ATmega328 chip, used in the second-generation Duemilanove and many other Arduino boards, has the following:

- **Flash memory** 32kB (2kB used by bootloader) for storing the program
- **SRAM** 2kB runtime memory, for dynamic variables, etc.
- **EEPROM** 1kB persistent storage (but limited lifespan).

In general computing, there are two schools of thought on how memory should be organised; the Von Neumann architecture (all memory is just memory) and the Harvard architecture (it's more efficient to split memory into pools – one for the program, and one for run-time memory the program will use. As it turns out, there are advantages and disadvantages on both sides. As a result, complex computers (like your desktop or even your phone) use a hybrid system – all memory seems to be just memory, but at a low level it is cached and organised in pools by the processor. The ATmega chips are designed for efficient, embedded and low-cost systems – there is no way complex memory management is going to be included, so you have a fairly straightforward Harvard architecture. The SRAM contains the runtime memory, the flash RAM contains the program memory.

However, 2kB of RAM is not really very much, and when you're dealing with arrays and such, it gets eaten up very quickly. Also, because of the way the memory stack and heap is organised and used, it doesn't even have to get full to cause you problems – it would be really painful to go into the details here, but basically 'freed' memory isn't always returned in a useful way, and when the stack and heap collide, all bets are off.

The solution to all of this is to try and be as memory efficient as possible. There are two particularly useful tricks you can use on the Arduino here:

PROGMEM

The first of these is to use the flash RAM of the Arduino to store variables in. This means that instead of using up your limited SRAM space for variables, you can store them in the substantially larger flash RAM instead.

There are a few caveats to this though. For a start, reading and especially writing values to the flash RAM is a slow process, partly due to the different types of storage, but more to do with the extra hoops that have to be jumped through to access memory that is outside of the architecture's stack. So, the flash RAM storage is not suitable for everything – loop counters, comms buffers and things that need to change regularly and be both read and written to are not good things to store in flash. Big lookup tables that you only rarely need to lookup, or other arrays of data that only need to be read infrequently are ideal for flash RAM.

In order to use flash RAM, there is a macro defined that takes care of storing data where you want it – you just need to add the keyword **PROGMEM** at the end of your variable definition (if you are using pre-1.5 versions of the Arduino IDE, you may need to check the official documentation for your specific version), and include the relevant header. For example:

```
#include "Arduino.h"
#include <avr/pgmspace.h>
 const char fred[8] PROGMEM = {
0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00
}
```

As you can see, just applying the **PROGMEM** keyword is enough. Theoretically, it can go anywhere on that line before the = sign. In practice, different versions of the IDE and the underlying compiler can throw up errors or strangeness. If you get any of these, just try moving the keyword to a different position.

Declaring your data is one thing; you also have to negotiate retrieving it. For example, to read this array back and do something to each byte, you would need to do the following:

```
char c;
for (char i=0; i<8; i++)
        {
    c = pgm_read_byte(&(fred[i]));
        myFunction(c);
        }
```

The **pgm_read_byte** function is one supplied by the library we included earlier, and is the only way to read a byte from the flash memory. To do so we need to supply it with an address, which we get with the construct **&( )**, which basically means "address of". So, by passing in the reference to the array, the function knows the address it needs to read data from. You can also iterate over this address in other ways, but remember that the difference between one data item and the next may vary depending on the size of what is being stored.

Writing to the array is not easily possible – flash memory needs to be written in large blocks, so aside from assigning values at runtime, you are pretty much stuck with it. Thankfully, for a lot of the stuff we would like to use it for – storing fonts, static arrays, maybe lookup tables – all the big users of storage, we don't need to alter the contents.

F()ing strings

Another handy macro is the **F()** function. One of the biggest memory hogs is storing string values. Each character takes a byte, then there is a stop byte at the end. You would be surprised how much memory they can take up. Strings though are ideal candidates to store in flash memory, at least the kind of message strings that we might use.

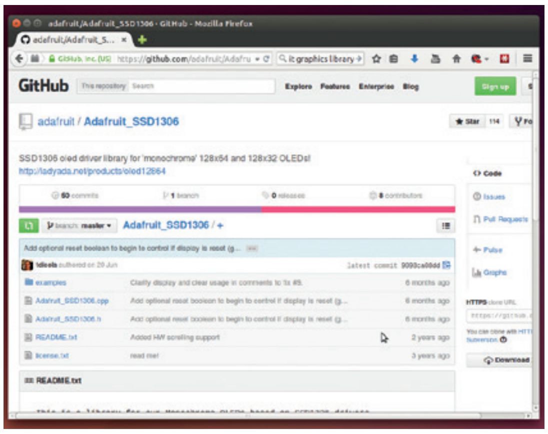
Fortunately, there is a macro for this too. Just wrap the string in **F()** and it will be magically stored in flash memory and fetched from there when required. You can easily see this in action with the following examples; enter this as a complete program in the Arduino IDE and press the tick button to compile it:

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(1000);
  Serial.println("Hello, I am a string.");
}
```

now replace the serial output line with:

```
Serial.println(F("Hello, I am a string."));
```

...and press the tick again. If you check the compiler output, you will see the second example uses 22 bytes less "dynamic memory" (SRAM) and 30 bytes more of "program storage space" (flash RAM). the extra bytes are due to the additional operations required to fetch the string from flash, but that's a pretty good exchange rate. Storing all your static strings like this can make a huge difference if you are running out of storage space.


adafruit/Adafruit_SSD1306 · GitHub - Mozilla Firefox

https://github.com/adafruit/Adafru

adafruit / Adafruit_SSD1306

SSD1306 oled driver library for 'monochrome' 128x64 and 128x32 OLEDs!
http://ladyada.net/products/oled12864

50 commits | 1 branch | 0 releases | 8 contributors

branch: master | Adafruit_SSD1306 / +

Add optional reset boolean to begin to control if display is reset (g…
ldeols authored on 20 Jun | latest commit 9093ca08dd

examples	Clarify display and clear usage in comments to fix #5.	6 months ago
Adafruit_SSD1306.cpp	Add optional reset boolean to begin to control if display is reset (g…	6 months ago
Adafruit_SSD1306.h	Add optional reset boolean to begin to control if display is reset (g…	6 months ago
README.txt	Added HW scrolling support	2 years ago
license.txt	read me!	3 years ago

README.txt

The Adafruit libraries (which are in general, excellent) are on GitHub too, and may be useful for further inspiration: **https://github.com/adafruit**.

This does assume the 'cursor' is in the correct place. This is simply a matter of setting up the addressing modes correctly :

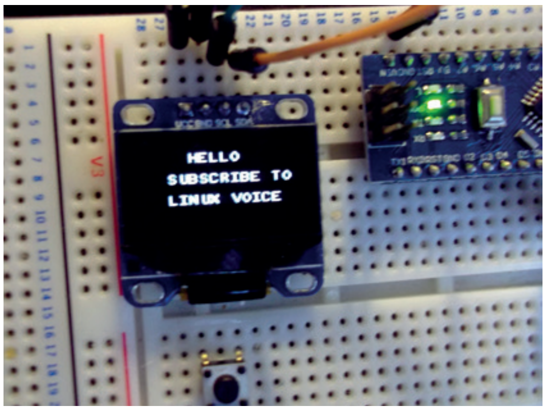
```
void evilOLED::setCursor(char x, char y)
{
        _row= (
        (y >7)
        ? 7
        : y
        );

        _col=(
        (x > 15)
        ? 15
        : x
        );

        sendCmd(0xB0+_row); //set page from row
        sendCmd(0x00+(_col & 0x01)*8);
        sendCmd(0x10+(_col>>1));
}
```

This may look like gibberish if you haven't used the ternary operators in C or C++ before, but it merely forces the x,y values to be 0–15 and 0–7 respectively, before doing some shift operations to set the row and page address.

GitHub is great

Even if you are working on a particular library purely for yourself, there are great advantages to using some sort of online version control system. For a start, you will always know where the latest version is, even if your local copies get deleted by solar storms or eaten by the dog. And though you may believe that you and only you are interested in your library, you may be surprised that other people are willing to help out and add stuff/fix bugs too.

GitHub is one of the most popular online versioning systems, even if a lot of the *Git* commands might be strange and confusing. It is completely free for open source projects, and is great for Arduino libraries (you will find a few on there), if only because it conveniently allows you to download the latest version as a Zip file with one click (the Arduino IDE can automagically add Zip files).



If you run out of ideas, you can always sell advertising space…

To go further than this, it is more useful to have a routine to print an actual string, rather than a character at a time. This is also in the source listing – it merely adapts the above code to work from a string pointer, looping through and printing each character.

Further, we can also overload this function to print integers too, by first converting to a string:

```
void evilOLED::putString(int s)
{
        char buffer[16];
        itoa(s, buffer,10);
        putString(buffer);
}
```

Wrapping it up

So, we have seen how we can manipulate bytes into fonts, bytes into images and save a lot of precious memory in the process. We have also seen how to do some hardware enablement and write a functioning Arduino library. The overriding point here is that if you are willing to get your hands dirty with some (quite complicated at times) manipulation of bits and bytes, you can create a custom implementation of something that exactly suits your needs.

This code can be extended in so many ways – you could just create different fonts for different scenarios. It would also be possible to display the fonts double-size for example (just write each pixel twice in each direction) or even invert them (invert each byte before you write it).

And much though we weren't keen on framebuffers, you could maybe create a framebuffer for part of the screen, say the last two lines or whatever. If you were cunning, this could be allocated on demand so it didn't waste memory if it wasn't used. The possibilities are limited only by the laws of physics and the endurance of your typing fingers.

The complete code, including the utilities and sample font and splash screen, are conveniently located on GitHub for your delight. And mine. If you come up with some useful variants (or find any bugs!) please fork or contribute : **https://github.com/evilnick/evilOLED**

Nick Veitch has edited computer magazines for 1,000 years. He now works at Canonical and collects gin bottles.

LINUXVOICE TUTORIAL

BUILD A WEB BROWSER WITH 20 LINES OF PYTHON

GRAHAM MORRISON

Drag and drop your user interface and tie it all together with a few simple lines of Python.

WHY DO THIS?

- Learn how to quickly prototype ideas.
- Develop rich GUI applications with a little Python.
- Delve into the *Qt* API.

The *Qt* graphical toolkit has been at the heart of the KDE desktop since its inception, and it's used by many other cross-platform applications. It's a great because it does so much of the hard work for you, even at a low level. There's a *Qt* class for dealing with string manipulation, for example, or sorting lists. There's exceptional networking support and transparency, file handling, native XML and image handling. Using *Qt* to perform all these tasks means you don't have to re-invent the wheel or import yet another library into your project. But *Qt* is still best known for it's high level user-interface design, where you can quickly construct an application from buttons, sliders, forms and images and tie them all together from your code.

Most developers have always used C++ to develop their *Qt* applications, but recent years have seen user-interface designers embrace *Qt*'s native QML language for adding non-API functionality without the formality of a C++ build environment. QML is much like JavaScript and enables you to quickly fix components together. It's the magic behind the new widgets in KDE 5, for instance, hopefully allowing lots more people to quickly add functionality to their desktops. But there have always been other options too. In particular, and the focus for these two pages, there are Python bindings provided by two separate projects – *PySide* and *PyQt*. Unfortunately, *PySide* development has slowed to a snail's pace and the project hasn't been able to support *Qt 5*. Which leaves us with *PyQt*. It's a brilliant open source implementation that's slightly less liberal than PySide, perhaps because of a commercial version, but it offers a great community and documentation. And because it's still open source, you can install it from almost any distribution. What we're going to do with just 20 lines of Python is create a fully interactive web browser, hopefully showing Python and *Qt* are a brilliant match for quick and easy application development, with all the advantages of both Python and *Qt*.

Get coding

One of the best things about developing *Qt* Python apps is that you don't need a build system and you don't need to compile anything. You could even use the Python interpreter if you wanted to – typing commands and seeing the results in real time. To get started, you'll need *Qt 5* installed, alongside the **PyQt5** packages. We're using version 3.4.2 of Python. You should also make sure you've got the package that includes the *pyuic5* utility, as we'll be using this to generate Python from the user interface GUI designer for added power. With all of that added to your distribution, it's time to create some code.

To illustrate how easy it is to write a Python/*Qt* application, we'll start off with a very simple and self-contained web browser that loads a specific page. Launch a text editor. The first thing we need to do is import the bits from *Qt 5* that we're going to need – just insert the following lines in the top of a new file:

```
from PyQt5.QtCore import QUrl
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWebKitWidgets import QWebView
import sys
```

How do you know which parts you're going to need before you've written the code? You don't. Normally when programming something like this you add to the 'from' section as and when you need to add components. In the above three lines, we're importing the ability to handle a *Qt* datatype called *QUrl*. As you might expect, this is a type that holds a URL, or a location on the internet. The reason this is its own type and not a text string is that *Qt* is able to test the value to make sure it's valid, and add functions to the data type so the programmer can do other things with the URL. Exactly what can and can't be done can be discovered from the *Qt* API documentation, which although it's written for C++, is just as relevant to *PyQt* as all the methods and types are the same. QUrls has functions to return a plain string or a filename, for instance, and it's the same with everything else you import from Qt. The other two components we import here are for the GUI. *QApplication* is the main application class for the GUI and its associated function and *QWebView*, as you might expect, is a *Qt* widget that uses *WebKit* to display websites. Finally, **import sys** adds a selection of system functions – we'll be using one of these to parse command line arguments. Here's the final section of code to add:

```
app = QApplication(sys.argv)
view = QWebView()
view.show()
view.setUrl(QUrl("http://linuxvoice.com"))
app.exec()
```

With the above short piece of code we create a fully functional web browser, albeit one where you can't manually enter your own URL. The first line creates the new application and window instance, passing the command line arguments for politeness (they're not used). After this, we instantiate the web view widget and assign this to **view**. This is made visible with the **show()** function, and updated with our own URL in the following line. The use of *QUrl* like this is known as casting because, we're using *QUrl* to force format a text string **http://linuxvoice.com** into a *QUrl* type because that's the only type accepted by the *QWebView* widget. After this, we run the application.

Save this file with the **.py** extension and switch to the command line. You can run the code by preceding the name of the file with the word **python** or **python3**. You should see a window appear and, as long as you've got an internet connection, a few moments later you'll see our web page. You can now navigate the site just as you would with any other browser.

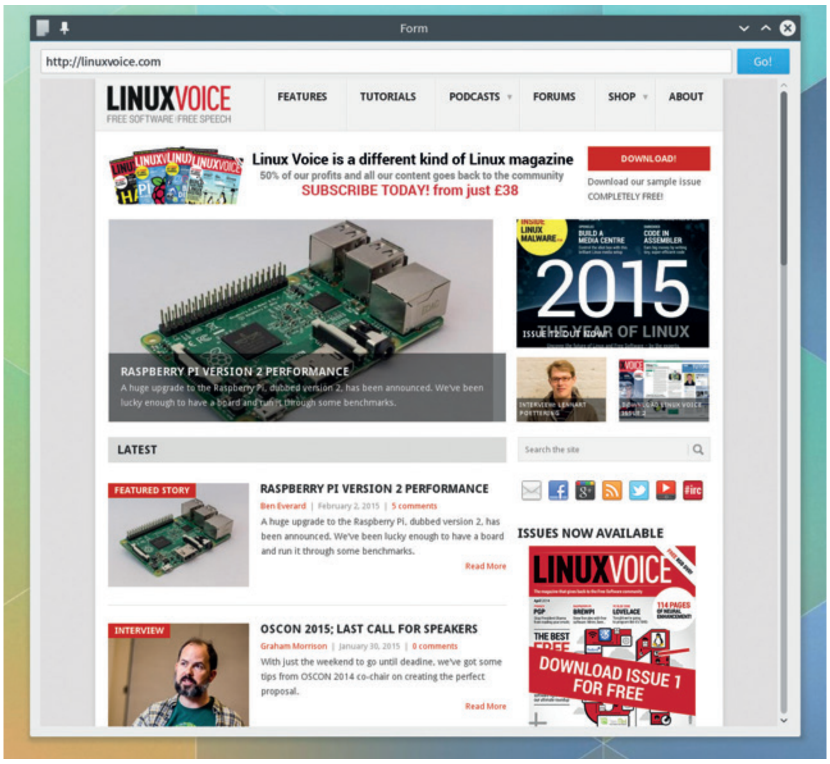
GUI designer

The biggest problem with our program is that you can't enter your own URL. We're going to solve this by adding both a URL input box and a button that tells your application to load the URL. But to make this more interesting, we're going to design this user-interface with one application, export the design and convert it into Python and then write another small script to add a little functionality. The application we're going to use to design the GUI is *Qt Designer*, and you should already have this installed as part of *Qt*.

Launch *Qt Designer* and from the startup wizard select a new design using QWidget as a base class. This will open an empty window into which you want to drag the QLineEdit, QPushButton and QWebView widgets. Layout in *Qt* is a little weird to get your head around, but to create a dynamically scalable version of a layout, place them into rough positions and select the Layout Using Grid option from the Form menu. Layouts are usually a careful balancing act that involves the horizontal and vertical grouping of objects that are again grouped together. We didn't change the name of any of the widgets created, nor the name of the main Form object. Usually you'd want to make these more descriptive.

Save the project and make a note of where the **.ui** XML file, is stored – ideally in the same place you're going to write your next Python script. Go back to the command line and navigate to that location. Now type **pyuic5 input.ui > ui_output.py**, changing the input and output names to fit yours. We'd recommend keeping the **ui_** prefix though, as we'll refer to this in the code. The **pyuic5** command will cleverly turn your user interface file into a Python script – take a look at the contents of the file to see what it's done. We're going to create a new script that inherits its properties from this and uses the form you create as the basis for the application.

The first section of our previous code needs a couple of additions. Firstly, QWidget need to be added after QApplication, because we use this in our inherited class. Secondly, we need to import the Python file generated by the user interface:



Linux Voice is a different kind of Linux magazine
50% of our profits and all our content goes back to the community
SUBSCRIBE TODAY! from just £38
Download our sample issue COMPLETELY FREE!

RASPBERRY PI VERSION 2 PERFORMANCE
A huge upgrade to the Raspberry Pi, dubbed version 2, has been announced. We've been lucky enough to have a board and run it through some benchmarks.

RASPBERRY PI VERSION 2 PERFORMANCE
Ben Everard | February 2, 2015 | 5 comments
A huge upgrade to the Raspberry Pi, dubbed version 2, has been announced. We've been lucky enough to have a board and run it through some benchmarks.

OSCON 2015; LAST CALL FOR SPEAKERS
Graham Morrison | January 30, 2015 | 0 comments
With just the weekend to go until deadline, we've got some tips from OSCON 2014 co-chair on creating the perfect proposal.

As *Qt* uses *WebKit*, your browser will be able to access almost any modern website.

```
from PyQt5.QtWidgets import QApplication, QWidget
from ui_output import Ui_Form
```

The next step is to add a block of code to manage our new class that inherits the user interface:

```
class MainWindow(QWidget, Ui_Form):
 def __init__(self, parent=None):
  super(MainWindow, self).__init__(parent)
  self.setupUi(self)
  self.pushButton.clicked.connect(self.pressed)

def pressed(self):
  self.webView.setUrl(QUrl(self.lineEdit.displayText()))
```

All we're doing here is creating a class we're calling **MainWindow** that inherits its layout properties from **Ui_Form**, which was imported from the user interface file we converted earlier. We then define the function that handles its initation. The crucial line is **self.pushButton.clicked.connect** because this is utilising *Qt*'s SIGNAL/SLOT mechanism to call a function called **pressed** when we click the button in the user interface. You can check what SIGNALs and SLOTs are supported by *Qt*'s functions from the API documentation. Following this, we write the short **pressed** function connected to the clicked event. This simply sets the **QUrl** used by the web view to the contents of the **lineEdit** widget. The web view automatically reloads when it gets this signal.

Finally, The only change to the main bit of code we need is to make our main view use our new class rather than **QWebView**:

```
view = MainWindow()
```

Saving and running this will now give you a very functional web browser and also fill you with ideas of how to use this for rapidly developing *Qt* applications with Python.

LINUX VOICE TUTORIAL

ASMSCHOOL: CONDITIONS, LOOPS AND LIBRARIES

MIKE SAUNDERS

Part 2: Start to write full programs and create your own library of useful subroutines.

WHY DO THIS?
- Learn what compilers do behind the scenes.
- Understand the language of CPUs.
- Fine-tune your code for better performance.

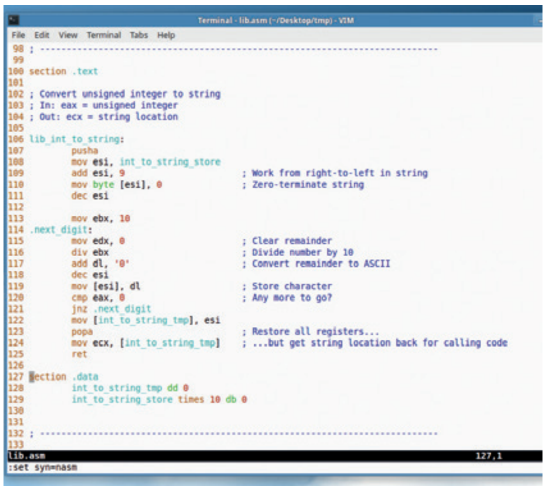
Last month we took our first steps along the path of assembly language programming, and saw that it's not as daunting as you might imagine. Yes, it feels rather alien when compared with high-level languages, and you don't have lots of fancy data types and levels of abstraction to protect you from juggling memory directly. But that's what we love about assembly – it's a very pure form of coding, letting you speak to the CPU and operating system directly, without all that other fluff getting in the way. This issue we'll look at program flow, including loops and conditions, to help you create more powerful programs. We'll also show how to make your own subroutines to automate common tasks, and build up a library of useful code chunks that you can use in other programs. So, dim the lights, choose a retro green-on-black theme for your terminal emulator, and let's start hacking like in the good old days...

1 REUSE CODE WITH ROUTINES

It's perfectly possible to make modular and reusable code in assembly language. You have to be careful that you don't overwrite data used elsewhere in the program, but with the right approach you can create little black-box routines that you write once and never have to delve into again – you just call them when needed. For instance, in last month's program we used the kernel to print a text to the screen. If your program does lots of text printing, it could be a chore (and a waste of space) to set up the registers each time – if you remember, you have to put the system call number in **eax**, the output stream in **ebx**, the length in **edx** and so forth.

So let's move the string printing code into its own subroutine that we can call whenever we want. But! Let's also make it easier to use, in that we don't even need to specify the length of the string in the **edx** register. Our new subroutine can work that out itself.

First of all, create a variant of last month's sample program like this, and save it as **test.asm**:



```
Terminal - lib.asm (~/Desktop/tmp) - VIM
File  Edit  View  Terminal  Tabs  Help
; -------------------------------------------------------------------------------

section .text

; Convert unsigned integer to string
; In: eax = unsigned integer
; Out: ecx = string location

lib_int_to_string:
        pusha
        mov esi, int_to_string_store
        add esi, 9                      ; Work from right-to-left in string
        mov byte [esi], 0               ; Zero-terminate string
        dec esi

        mov ebx, 10
.next_digit:
        mov edx, 0                      ; Clear remainder
        div ebx                         ; Divide number by 10
        add dl, '0'                     ; Convert remainder to ASCII
        dec esi
        mov [esi], dl                   ; Store character
        cmp eax, 0                      ; Any more to go?
        jnz .next_digit
        mov [int_to_string_tmp], esi
        popa                            ; Restore all registers...
        mov ecx, [int_to_string_tmp]    ; ...but get string location back for calling code
        ret

section .data
        int_to_string_tmp dd 0
        int_to_string_store times 10 db 0


; -------------------------------------------------------------------------------

lib.asm                                                                    127,1
:set syn=nasm
```

Our **lib.asm** library will grow to be a useful resource of code snippets, such as number-to-string conversion.

```
section .text
        global _start

_start:
        mov ecx, mymsg
        call lib_print_string

        mov eax, 1
        mov ebx, 0
        int 80h

section .data
        mymsg   db 'Pretty cool, huh?', 10, 0

        %include "lib.asm"
```

This is similar to last month's code, but the string printing part has changed. Instead of setting up all the registers and calling **int 80h**, we just put the string location in **ecx** and then "call" a subroutine called **lib_print_string**. A "call" is a bit like a **GOSUB** in the basics of yesterday – it hands control to another routine, which will do its work and then return back into the main program.

Now, whereabouts does this **lib_print_string** routine live? Well, we haven't written it yet, but we're going to place it in **lib.asm**, another assembly language source file. In the listing above, you can see the **%include** line at the bottom, which simply adds the contents of **lib.asm** to the current code listing during the assembly phase. So, you can write commonly accessed routines in **lib.asm** and keep them separate, without them clutting up the main part of your program.

Another important thing to note here is the string of bytes next to the **mymsg** label. This time, as well as appending a 10 for a newline character, we've also added a zero. This turns it into a "null-terminated string" – and we can look for this zero in our code to determine the string length.

Building up the toolbox

Let's create the **lib_print_string** routine. This is longer and more complicated than the process we used last month, which may leave you thinking: what's the point? Why not just do it manually each time? Well, if you end up with a large program that has hundreds of string-printing parts, the overall code will be smaller if you use the same subroutine each time, rather than setting up all the registers manually. And as we mentioned earlier, this routine can work out string lengths too.

Put this code into **lib.asm** (you'll also find it in **www.linuxvoice.com/code/lv013/lib.asm** with some other routines you may find useful):

```
section .text

; Print text string
; In: ecx = string loc
; Out: Nothing

lib_print_string:
        pusha                   ; Save all regs

        mov eax, ecx            ; Save ecx for later
        mov edx, 0              ; Character counter
.loop:
        cmp byte [eax], 0       ; Is it zero?
        je .done                ; Jump ahead if so
        inc edx                 ; Increment counter
        inc eax                 ; And string loc
        jmp .loop               ; And carry on

.done:
        mov eax, 4              ; sys_write
        mov ebx, 1              ; stdout
        int 80h                 ; Call kernel

        popa                    ; Restore all regs
        ret                     ; Back to caller
```

There's quite a lot going on here, but it introduces various new concepts such as loops and conditionals, so we'll go through it carefully. One important point to note at the start: semi-colon characters are used to denote comments, so anything after them (until the end of the line) will be ignored by NASM. In assembly language, it's a very good idea to be verbose with your comments, otherwise you may come back to the code in several months and be completely bamboozled.

So, we start this off by telling NASM that the following code should be in the "text" section of the resulting binary file – that is, executable code and not data. Then we have three lines of comments, saying what the subroutine does, which registers it uses, and which registers it changes when it exits. We can see here that our **lib_print_string** routine just needs the string location in **ecx**, and doesn't change anything else – the registers will remain the same when control is handed back to the main program.

What is hexadecimal?

We use the base 10 (decimal) number system because that's how many fingers we've got, so it's useful for counting. But it doesn't make much sense in terms of a CPU, so in low-level programming you'll often see base 16 instead – aka hexadecimal. This takes some time to get your head around if you've never used it before, but after a while you can switch your brain into hex mode.

Like decimal, hex digits go from 0 to 9. But for 10 decimal, hex switches to A and then counts up to F (15 decimal). Then, for 16 decimal, it carries over – 10 in hex. Makes sense? This chart should help:

Dec:	1	2	3	4	5	6	7	8
Hex:	1	2	3	4	5	6	7	8
Dec:	9	10	11	12	13	14		
Hex:	0	A	B	C	D	E		
Dec:	15	16	17	18	19...			
Hex:	F	10	11	12	13...			

Hexadecimal 19 is followed by 1A, and FF is followed by 100, and so forth.

Other registers

Along with the four main general-purpose data registers, **eax**, **ebx**, **ecx** and **edx**, there are a few more that are worth knowing about. Two are used primarily for string handling: **esi** and **edi**. The first can be used as a "source index" for strings – ie a pointer to a position inside a string that's being read from – while the latter is a "destination index" for storing data. Consider this code:

```
mov esi, mystring
mov edi, blankstring
lodsb
stosb
```

Say that **mystring** points to a string containing "Hello" here, and **blankstring** just to a series of zeroed-out bytes. The **lodsb** instructions retrieves a byte from the location **esi** points to, storing it in **eax** (specifically, the **al** byte portion of that register), and then the **stosb** instruction stores the byte in **ah** at the position pointed to by **edi**. So it copies "H" from one string to another. But! To make string handling even easier, **lodsb** automatically increments the **esi** register each time by a byte (thereby pointing it to the next character), and **stosb** does the same thing for **edi**.

Meanwhile, the **esp** register points to the current location on the stack. This moves around as you push items onto it and pop them off. The **eip** register is the "instruction pointer" – it simply points to the current instruction in the code. This is changed when you do a jmp or call operation. And then there's the **EFLAGS** register, also known as the status register, which has various bits set to show the results of operations (eg whether the result was zero, or there was an overflow). This is used by many conditional instructions.

2 LOOPS AND CONDITIONALS

The first instruction in the previous block is "pusha", which means "push all registers onto the stack". You may remember the stack from last issue: it's a temporary storage space where you can place register contents when you need to use those registers for something else. We push all of the registers on to the stack at the start of the subroutine, do our own work with them, and then pop them all back off (with **popa**) just before we return back to the main code (with **ret**). This means that the calling program doesn't need to save the registers – it assumes they will be in the same state after the subroutine has been executed.

Now, the string location has been provided in the ecx register, but we want to keep that for later. We need to count the characters in the string though, so we copy ecx into eax and work with the former, leaving the latter well alone. We're going to go through the string until we find a zero byte (remember, we're using zero-terminated strings), counting up along the way to determine the string length. And the counter we're going to use is the **edx** register. In pseudocode:

```
10 Look at byte in string
20 Is it a zero?
30 If yes, exit the loop
40 If not, increment the counter and string position
50 Goto 10
```

Our loop begins with the **.loop** label, and the period at the start means that it's a local label. In other words, it expands to **lib_print_string.local**. Why do we need this? Well, it means we can use **.loop** as a label in other routines, which is very handy – otherwise you'd need to come up with a different name each time. Of course, you can still only have one instance of **.loop**

In the next couple of issues, we'll prepare for running code on bare hardware – no OS required!

Do the math

So far we've focused on moving numbers into registers, but we can also perform mathematical operations on registers as well. For example, here's some addition:

```
mov eax, 10
mov ebx, 15
add eax, ebx
add eax, 7
```

Remember that operations go from right to left in assembly language (at least, in NASM syntax). So what does **eax** contain by the end? First of all we place 10 into it, and then 15 into **ebx**. We add **ebx** on to **eax**, so the latter now contains 25. Then we add a number (7) so the result is 32. Subtraction can also use registers and numbers:

```
mov eax, 100
sub eax, 99
```

Now **eax** just contains 1. Multiplication and division work in a slightly different way – you'll find a lot of these oddities in x86 programming, due to the long history of the architecture! For multiplication, you first need to place a number in **eax**; then you multiply it using another register. For instance:

```
mov eax, 10
mov ebx, 5
mul ebx
```

After this, **eax** contains 50. Division works in a similar way, with the remainder being stored in the **edx** register:

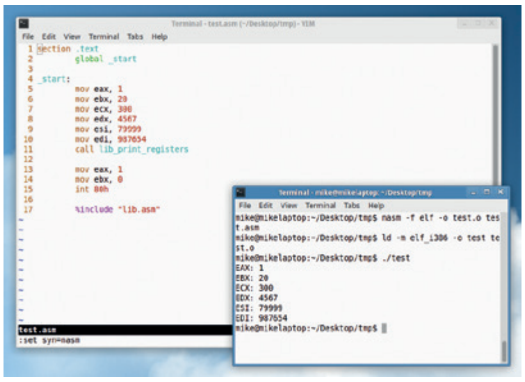
```
mov eax, 10
mov ebx, 4
div ebx
```

After this, **eax** contains 2 (as there are two fours in 10), and **edx** contains 2 as well, as that's the remainder.

Note that you can't use **lib_print_string** to print the contents of registers directly, as they need to be converted into ASCII text format first. This is beyond the scope of this tutorial, but if you download **www.linuxvoice.com/code/lv013/lib.asm** you'll see there are two extra routines: **lib_int_to_string** (which takes a number in **eax**, and returns the location of a string with the converted form in **ecx**), along with **lib_print_registers**, which simply shows the contents of all registers. So, try doing some maths and then:

```
call lib_print_registers
```

to see what the results are.


Terminal - test.asm (~/Desktop/tmp) - VIM
File Edit View Terminal Tabs Help
Terminal - mike@mikelaptop: ~/Desktop/tmp
File Edit View Terminal Tabs Help

```
section .text
        global _start

_start:
        mov eax, 1
        mov ebx, 20
        mov ecx, 300
        mov edx, 4567
        mov esi, 79999
        mov edi, 987654
        call lib_print_registers

        mov eax, 1
        mov ebx, 0
        int 80h

        %include "lib.asm"
~
test.asm
:set syn=nasm
```

```
mike@mikelaptop:~/Desktop/tmp$ nasm -f elf -o test.o tes
t.asm
mike@mikelaptop:~/Desktop/tmp$ ld -m elf_i386 -o test te
st.o
mike@mikelaptop:~/Desktop/tmp$ ./test
EAX: 1
EBX: 20
ECX: 300
EDX: 4567
ESI: 79999
EDI: 987654
mike@mikelaptop:~/Desktop/tmp$
```

Use **lib_print_registers** from our **lib.asm** to quickly view the contents of the main registers.

under each parent label.

So, we have the string position in **eax**. The first thing we do is to get a byte from the string like so:

```
cmp byte [eax], 0
```

The **cmp** instruction means "compare one number (or register) with another", and the square brackets are extremely important here. They mean: don't compare the number inside **eax** to zero, but the byte inside the memory location that **eax** points to. You see, **eax** will contain a big number pointing to a string somewhere in memory, like 2187612. We're not interested in that location, but we're interested in the exact byte stored inside that location, which is why we're using the square brackets. So here we're comparing a byte in the string to zero. The next line, **je .done**, is a conditional jump operation: it says, "if the numbers are equal, jump to the specified point in the code". There are other conditional jump operations that you can do after a **cmp**, such as **jg** (jump if first number is greater than second), which works with signed numbers, or **ja** (jump if above) which works with unsigned ones. We'll look at these in more detail next month.

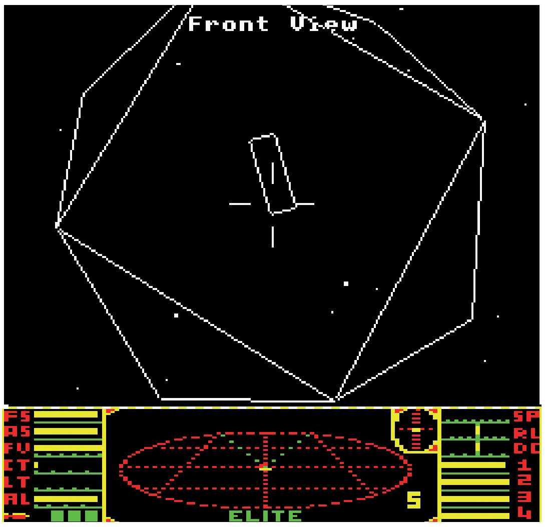
Back to the code: if the byte from the current position in the string (pointed to by **eax**) is zero, it means it's the end of the string, so jump ahead to the **.done** label. But if it's not zero, the lines ahead of the **je** instruction are executed instead. First we increment **edx**, our counter, by one byte, and then we increment **eax** so that it points to the next character in the string. Then we jump back up to our **.loop** label, and perform the next comparison.

Do more with less

As we've mentioned before, this is exactly how CPUs work: they move numbers around between memory and registers, perform calculations on them, and then jump to different parts of the code accordingly. Assembly language is actually very simple in some ways – it just takes time to build up advanced functionality from a relatively small set of instructions. Look at *Elite*, the 8-bit space trading classic, for instance: it was originally written for a CPU (the 6502) that had only 56 instructions. Yet with this small range of instructions, it's possible to create a complete 3D engine and an absorbing game around it!

Anyway, at the **.done** label in our code, we have everything we need to call the kernel. We need to put the value 4 into **eax** to specify the **sys_write** kernel routine, and 1 into **ebx** to print the string to standard output, but we've already populated the other registers accordingly. We left **ecx** alone since the start of the routine, and our loop has calculated the length of the string in **edx**, so the kernel can use that too! So all we need to do is call **int 80h** and the job's a good 'un, as they say.

Then we pop the registers back off the stack, as described earlier, and use **ret** to return to the calling program. There's something important to note here: when you "call" a different piece of code, the current location is placed onto the stack for later retrieval. The **ret** instruction then retrieves that location from the stack and places it inside the **eip** (instruction pointer) register, so execution continues from the previous point. This is why you have to be careful when managing the stack: if you pop off more numbers than you've pushed on, you could end up popping off the return address from a call instruction, and your program will end up executing somewhere else!



Early CPUs with tiny sets of instructions could still do a lot – look at *Elite* on the 6502 for instance.

"Assembly language is very simple – it just takes time to build up advanced functionality."

You can assemble and link the program with the same instruction as last month:

```
nasm -f elf -o test.o test.asm
ld -m elf_i386 -o test test.o
```

Then run the program in place with **./test**. Yes, it prints a string, just like last month! But it does much more under the hood, and will prepare you for building up a library of useful routines in the future.

You now know how to repeat operations using loops, which is vital in more advanced programs, and you can also perform conditional operations depending on the contents of a register. You can also create your own subroutines to handle common tasks, and make them sufficiently modular that they won't interfere with the workings of the calling code. (This is why it's a good idea to use **pusha** at the start and **popa** at the end of a subroutine, when the subroutine works with several registers – you can guarantee the calling code that everything will be in the same state when you **ret** back to it.

Next issue we'll look at input and handling files, so you'll be ready to write proper, functional (and blisteringly fast) programs. And we'll get even closer to the ultimate goal of running code on the bare metal of your PC. See you then... LV

Mike Saunders has written a whole OS in assembly (http://mikeos.sf.net) and is contemplating a Pi version.

LINUX VOICE MASTERCLASS

You wouldn't want other people coming into your machine and poking around – so set up a firewall!

FIREWALLING MADE EASY

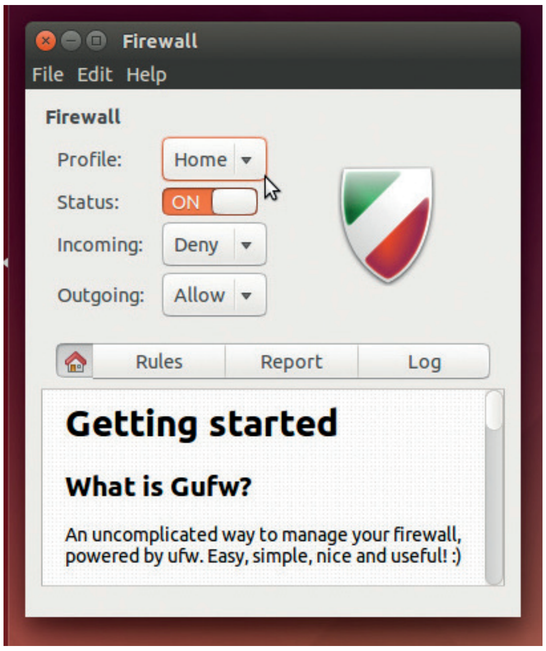
Use the kernel's network filter to block unwanted network traffic.

JOHN LANE

A firewall is a security system that gives you control over data that applications running on your computer may send and receive across any networks it's connected to. This network traffic is sent and received in chunks called "data packets" that can be inspected and filtered according to rules, a technique known as "packet filtering".

The Linux kernel has, since its early versions, had a built-in packet filter. The current implementation, introduced with version 2.4 of the kernel, is called Netfilter and, along with its command-line *iptables* tool, provides the foundation for firewalls on Linux. Earlier kernels used now-superseded tools called *ipfwadm* and *ipchains*.

The easiest way to protect your computer is to use a firewall application. These sit above and use *iptables* so you don't have to. They provide a more user-friendly way to create and maintain firewall rules, often through a GUI interface. There are several such applications and, in this month's Masterclass, we'll take a look at a few of them.



Getting started

What is Gufw?

An uncomplicated way to manage your firewall, powered by ufw. Easy, simple, nice and useful! :)

Flick the status switch to on to start your firewall. It doesn't get much easier than *Gufw*.

A good way to get started is to keep things simple – something that *ufw* (the Uncomplicated Firewall), and its *GTK* application *Gufw* aims to do. In its own words, it's "an uncomplicated way to manage your firewall". It's part of the default Ubuntu installation but it's in other distributions' repositories too. You can head over to **gufw.org** for the latest.

You need privileged access to use *iptables*, so *Gufw* requires you to enter the root password. You can instead start it with **sudo** or while logged in as root.

The firewall is initially disabled and all network traffic is permitted. Switching the status to 'on' on the main screen enables a basic firewall that allows outbound network traffic but blocks anything coming in. You can change this behaviour using the 'incoming' and 'ouggoing' options at the top of the *Gufw* window. As firewalls go, it isn't really get any much simpler than that.

Let me in

You probably won't want to be quite that black-and-white, however, and this is where rules come in. Rules enable you customise the firewall's configuration. As an example, imagine we have another machine and want to use SSH to connect from it. With the firewall enabled, that's no longer possible – not until we add a rule to allow it.

Click on the 'Rules' button to display the rules page and then click on the plus sign to add a new rule. There are preset configurations for many applications and system services – use the available drop-downs to locate what you want or type a name in the Application Filter box. In our case, we want to allow SSH, so search for that. Also choose the appropriate direction and policy – one of "Allow", "Deny", "Reject" or "Limit" – we want to allow inbound SSH.

You'll see a warning that using a default Allow policy is a security risk. These helpful messages make *Gufw* a good choice for beginners. The default Allow policy accepts all connections, but it's more secure to restrict access to the machine that we'll SSH from. To do this, click on either the Advanced tab or the right-arrow button next to the search box to view the advanced options. Here you can specify the IP addresses to allow. Once you have your firewall configured, you can use the Listening Report to monitor it. This is on the Report tab and shows listening TCP and open UDP ports.

Beyond the basics

If you want a little more control, *fwbuilder* (the *Firewall Builder*), is another option that you can use to configure and manage *iptables* as well as firewalls on other platforms such as BSD, OS X and Windows as well as some dedicated firewall hardware appliances. It's a *Qt* application and is therefore a little heavier than *Gufw*. You should find it in your package repositories and you can go to **www.fwbuilder.org** for more information.

A key difference between *Firewall Builder* and *Gufw* is that you can use it to prepare firewalls to be installed on other computers or devices. You don't need privileged access to use it, so regular users can build firewall configurations; however, installing them does require such access to wherever they will be installed.

This build and install paradigm is one way that *fwbuilder* differs from *Gufw*, where changes take effect immediately.

When you launch *fwbuilder* you're offered the online quick start guide, which opens in your browser and introduces the main concepts. It's a much more comprehensive application than *Gufw*, so it's worth working through the guide. There's also a comprehensive user manual and tutorial videos on the website.

Firewall Builder uses "Objects" to represent the things you'll write rules for. They're grouped into libraries and there are two of these object libraries by default: one is called "User" and is where user-created objects are stored; the other one is called "Standard" and is a read-only library that contains predefined objects such as common TCP and UDP services.

You create a policy containing your firewall rules and compile this to convert your rules from the *Firewall Builder* syntax into the command syntax used by the target firewall platform (on Linux, that's *iptables*). The policy needs to be compiled and re-installed whenever any changes are made to it. You begin by creating a new firewall – click the button for this (it's got a picture of a brick wall on it). Give it a name, choose the *iptables* firewall software and (obviously) the Linux operating system. There's a checkbox that you can enable if you want to start with a preconfigured firewall; this is a good way to get started because the basic rules will be added for you.

When you're happy with your configuration, press the "install" button to compile and install it. This generates a shell script containing the *iptables* commands that implement your configuration and then uses SSH to copy the generated script to the firewall host and execute it (you'll need to enter credentials for SSH when requested).

Be persistent

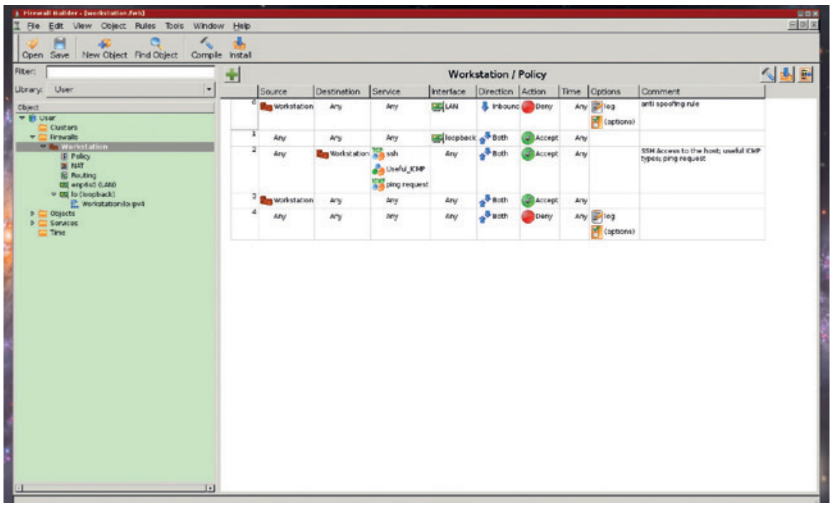
The kernel's netfilter state doesn't survive reboots, so you'll probably want your system to configure it each time it boots. How you do this will depend on your distribution and choice of firewall. *ufw* has a service that takes care of this for you – Ubuntu users can enable it with Upstart:

```
$ sudo ufw enable
```

A similar service is provided for Systemd. Distributions that use Systemd also have generic *iptables* services that load its configuration from a file stored in **/etc/iptables**. *Firewall Builder* makes it easy to use this without having to open a command-prompt. Double-click the firewall entry and then press "Firewall Settings" and select the "Prolog/Epilog" tab. Enter the following commands into the lower box; they will be added verbatim to the generated configuration to save and restore your settings:

```
systemctl enable iptables ip6tables
iptables-save > /etc/iptables/iptables.rules
ip6tables-save > /etc/iptables/ip6tables.rules
```

These tools make it easy to implement a firewall without needing to understand the inner workings of netfilter and *iptables*. *Gufw* is sufficient to protect your own computer but if you if you have many machines or just want more control, *fwbuilder* is the way to go.



Workstation / Policy

	Source	Destination	Service	Interface	Direction	Action	Time	Options	Comment
0	Workstation	Any	Any	LAN	Inbound	Deny	Any	log (options)	anti spoofing rule
1	Any	Any	Any	loopback	Both	Accept	Any		
2	Any	Workstation	ssh Useful_ICMP ping request	Any	Both	Accept	Any		SSH Access to the host; useful ICMP types; ping request
3	Workstation	Any	Any	Any	Both	Accept	Any		
4	Any	Any	Any	Any	Both	Deny	Any	log (options)	

Starting *Firewall Builder* with a preconfigured firewall is a good place to begin, and the drag-and-drop interface makes customising easy.

Firewall policies

The firewall policies determine the action when a packet matches a rule. A packet may be accepted, which lets it through, or not. In the latter case, the netfilter may or may not send a rejection message to the source; such packets are said to be dropped and the port appears closed to the source.

Unfortunately the firewall applications use different terminology. *Gufw* has 'policies' called Accept, Reject and Drop. In *fwbuilder* the so-called "Actions" are Accept, Reject and Deny as well as a few others for logging and custom functionality.

LV PRO TIP

KDE users may wish to try *kcm-ufw* as an alternative to *Gufw*. It's a KDE control centre module for configuring and controlling *ufw*.

A LOOK AT COMMAND-LINE TOOLS FOR NETWORK SECURITY

Build an uncomplicated firewall and pimp it up with iptables…

JOHN LANE

The "Uncomplicated Firewall" is the easiest way to create a firewall on your computer. We've looked at its GUI application, but that's just a wrapper around its command line tool, *ufw*, which you can use directly from the command line to quickly establish a firewall.

```
$ ufw enable
Firewall is active and enabled on system startup
```

The default behaviour is to allow outbound connections and block inbound ones. You can explicitly set these or other behaviours:

```
$ ufw default deny incoming
$ ufw default allow outgoing
```

Writing custom rules is also straightforward. We can allow inbound **ssh** like this:

```
$ ufw allow ssh
```

You'll want to list your rules. You can list them with their numeric ID, which makes deleting them easier:

```
$ ufw status numbered
Status: active
   To  Action    From
 1  22  ALLOW IN  Anywhere
```

You can then delete rules:

```
$ ufw delete 1
```

Writing more specific rules follows a straightforward syntax that is described on the manual page (see **man ufw** for a complete description). You can, for example, add a rule for a specific host:

```
$ ufw allow from 192.168.5.10 to any port ssh
```

Although *ufw* presents a very simple command line interface, it is capable of doing anything that netfilter can do because it uses the standard *iptables* configuration underneath, and you can modify this to meet your needs. Before we can do this, it's worth understanding *iptables* a little.

LV PRO TIP

Using *ufw* or *iptables* requires root privileges, so log in as root or use **sudo**.

LV PRO TIP

Use **ufw show added** to list your rules when the firewall is stopped.

Factory reset

To reset netfilter to its default state, you have to delete all custom rules and chains and return the built-in chains' default policies to "Accept". You can also reset its built-in counters to zero. There isn't a single reset command, but here is a short script that you can use:

```
#!/bin/sh
 [ $EUID == 0 ] || { sudo $0 "$@"; exit; }
for iptables in iptables ip6tables; do
  echo --flush --delete-chain --zero | xargs -n1 $iptables
  echo -n INPUT OUTPUT FORWARD | xargs -n1 -I% -d\
$iptables --policy % ACCEPT
done
```

It must be run while logged in as root.

Contents

Current Shorewall Releases
What is Shorewall?
Getting Started with Shorewall
Looking for Additional information?
Donations
License
Logo

Current Shorewall Releases

Current Stable Release	4.6.5.5	Release Notes	Known Problems
Prior Stable Release	4.5.21.10	Release Notes	Known Problems
Development Release	4.6.6 RC 1	Release Notes	

Read more about the Shorewall 4.5/4.6 releases here. Get them from the download sites

What is Shorewall?

Shorewall is a gateway/firewall configuration tool for GNU/Linux.

For a high level description of Shorewall, see the Introduction to Shorewall. To review Shorewall functionality, see the Features Page.

Getting Started with Shorewall

The *Shorewall* firewall is well documented and contains all the information you'll need to produce your ultimate firewall without having to learn *iptables*.

The kernel's netfilter configuration is a collection of "tables" comprising "chains" of "rules". There are four built-in tables: the Filter table is the default table for packet filtering rules and is what we'll focus on. The other tables are NAT and Mangle for packet alteration, and a Raw table exists for configuration exemptions:

- **INPUT** for inbound packets.
- **OUTPUT** for outbound packets.
- **FORWARD** for packets routed through the local server.

Each chain can contain zero or more rules and has a default policy that defines what to do with packets that don't match any of the chain's rules. The default policy is to accept.

Chains contain rules that are basically a condition and a "target" that describes what happens when the condition is matched. The main targets are ACCEPT and DROP, and only the first matching rule is applied. Rules should be ordered so that less-specific ones come first, and rules in a chain oppose the chain's default policy – a chain with a default DROP policy usually contains ACCEPT rules. Rules should be considered as exceptions to the default policy.

A good place to start getting to know the netfilter is to use *iptables* to list the current rules. A clean netfilter with no configuration would look like this:

```
$ iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source               destination
Chain FORWARD (policy ACCEPT)
target    prot opt source               destination
Chain OUTPUT (policy ACCEPT)
```

```
target     prot opt source
```

You can see the three chains we previously described, each with a default ACCEPT policy but no rules. We'll add some rules to prevent inbound connections except for SSH; it goes like this:

```
$ iptables -A INPUT -i lo -j ACCEPT
$ iptables -A INPUT -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT
$ iptables -A INPUT -p tcp --dport 22 -j ACCEPT
$ iptables -P INPUT DROP
```

This illustrates the more complex nature of *iptables*. We begin with some generic rules that are necessary for the general operation of the system. The first accepts any traffic on the loopback interface that processes use to talk to other processes on the same system. The next rule accepts packets that are part of an already established session. It uses a netfilter module called "conntrack" that inspects packets to identify those associated with such sessions (the netfilter architecture is extendable through modules in a similar fashion to the kernel and stateful packet inspection is implemented as a module). The third rule is what accepts new SSH connections, and the final thing we do is change the input chain's default policy so that it drops packets unmatched by our rules.

With these rules, the input chain now looks like this (**iptables -vL** for more verbose output):

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in  out  source    destination
    0     0 ACCEPT     all  --  lo  any  anywhere  anywhere
    0     0 ACCEPT     all  --  any any  anywhere  anywhere    ctstate
RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  any any  anywhere  anywhere    tcp
dpt:ssh
```

Any configuration that you perform using *iptables* is lost when you shut down or restart your system. These means that your firewall configuration needs to be applied each time your system boots. There are two tools provided to help you persist your netfilter configuration; one saves and another restores:

```
$ iptables-save > /etc/iptables/iptables.rules
$ iptables-restore < /etc/iptables/iptables.rules
```

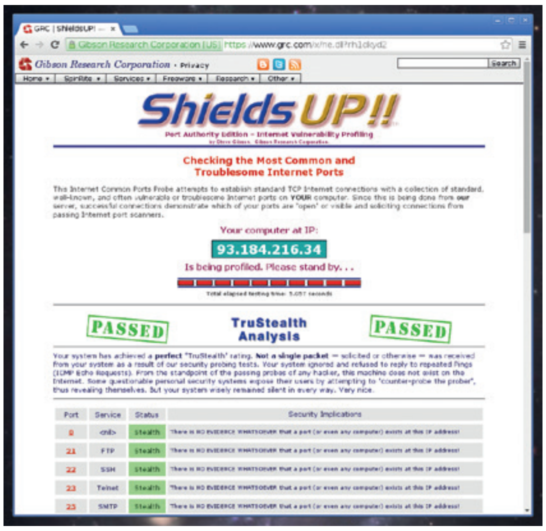
You still need to perform the restore process during reboot, and there isn't a consistent way to achieve this. Ubuntu users can look at the **iptables-persistent** packagem and those using Systemd can use its *iptables* service (this loads rules from the file we used in the **iptables-save** example):

```
$ systemctl enable iptables
```

Hacking ufw

If you look at the output of **iptables-save**, you'll see that it contains the command line arguments we used to create our rules. You can use this knowledge to write rule files directly, and this brings us back to *ufw*: its configuration files use the same **iptables-save** format and are stored in the **/etc/ufw** directory.

Like *iptables*, *ufw* has separate configurations for IPv4 and IPv6. Each has **before.rules** and **after.rules**


GRC | ShieldsUP!
Gibson Research Corporation [US] https://www.grc.com/x/ne.dll?rh1dkyd2
Gibson Research Corporation · Privacy
Search
Home
SpinRite
Services
Freeware
Research
Other
ShieldsUP!!
Port Authority Edition – Internet Vulnerability Profiling
PASSED
PASSED

Checking the Most Common and Troublesome Internet Ports

This Internet Common Ports Probe attempts to establish standard TCP Internet connections with a collection of standard, well-known, and often vulnerable or troublesome Internet ports on **YOUR** computer. Since this is being done from **our** server, successful connections demonstrate which of your ports are "open" or visible and soliciting connections from passing Internet port scanners.

Your computer at IP:

93.184.216.34

Is being profiled. Please stand by. . .

Total elapsed testing time: 5.057 seconds

TruStealth Analysis

Your system has achieved a **perfect** "TruStealth" rating. **Not a single packet** — solicited or otherwise — was received from your system as a result of our security probing tests. Your system ignored and refused to reply to repeated Pings (ICMP Echo Requests). From the standpoint of the passing probes of any hacker, this machine does not exist on the Internet. Some questionable personal security systems expose their users by attempting to "counter-probe the prober", thus revealing themselves. But your system wisely remained silent in every way. Very nice.

Port	Service	Status	Security Implications
0	<nil>	Stealth	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
21	FTP	Stealth	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
22	SSH	Stealth	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
23	Telnet	Stealth	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
25	SMTP	Stealth	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!

You can use a probing service such as ShieldsUP!! to check your firewall from outside your network.

files that are applied before and after your own *ufw* rules, which are stored in the same format at **/lib/ufw/user.rules** – look there to see the *iptables* arguments generated by your *ufw* commands.

As your knowledge of *iptables* develops, you can use it to customise your *ufw* rules while also retaining the benefits of its easy-to-use command line tool. You win both ways.

The ultimate Linux firewall?

If you need more than what *ufw* offers yet see *iptables* as a step too far, or if you need to configure firewalls for multiple computers, another popular choice is the *Shorewall*, command-line netfilter configuration tool that can meet the needs of users with more complex requirements. It has a comprehensive website (**http://shorewall.net**) where there's lots of documentation including a Getting Started guide. There's documentation for various scenarios, and its "Universal Configuration" provides a default firewall similar to *ufw*. To use this basic *Shorewall* firewall configuration, after installing the package from your distro's repository, copy the configuration into place and start it:

```
$ cp -a /usr/share/doc/shorewall/Samples/Universal/* /etc/
shorewall
$ shorewall start
```

The lowdown

All the firewall tools that we have looked at are built on top of *iptables*. They all create netfilter rules and apply them using *iptables*, but they present a higher level of abstraction to the user that hides its complexity. While the intention is to simplify things for the user, a disadvantage of this approach is that some netfilter capabilities may not be available.

Whichever of these tools you eventually adopt, you'll be better placed to implement secure systems. LV

John Lane provides technical solutions to business problems. He has yet to find something that Linux can't solve.

LV PRO TIP

There are completely separate netfilters for IPv4 and IPv6, and separate *iptables* and *ip6tables* commands to manage them. Remember to use the appropriate tools when using either or both protocols.

NEXT MONTH IN LINUX VOICE

ON SALE THURSDAY 26 MARCH

COMPLETE GUIDE TO HACKING

ETHICAL HACKING

Learn how the bad guys work and use that knowledge to protect yourself. Starring Ben Everard and the Metasploit framework.

EVEN MORE AWESOME!

Eben Upton
The benign overlord of the Raspberry Pi Foundation gets us all excited over what's to come after the Raspberry Pi Version 2 fuss has died down.

Ubuntu phones
We've been looking forward to it for ages, and now the first smartphones to ship Ubuntu's phone OS are with us at last. Here's how they measure up.

Inside x86
Explore the inner workings of the most common PC architecture – the ubiquitous x86 – with our kernel superstar Dr Valentine Sinitsyn.

LINUX VOICE IS BROUGHT TO YOU BY

Editor Graham Morrison
graham@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Technical editor Ben Everard
ben@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Creative director Stacey Black
stacey@linuxvoice.com

Editorial consultant Nick Veitch
nick@linuxvoice.com

All code printed in this magazine is licensed under the GNU GPLv3

Printed in the UK by
Acorn Web Offset Ltd

Disclaimer We accept no liability for any loss of data or damage to your hardware through the use of advice in this magazine. Experiment with Linux at your own risk!
Distributed by Marketforce (UK) Ltd, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel: +44 (0) 20 3148 3300

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

ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**

/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!

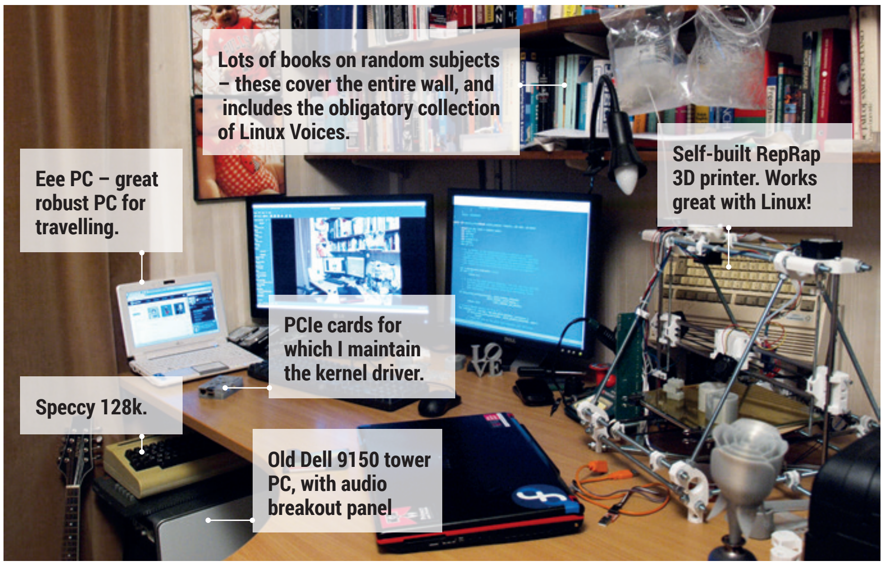
I have fallen out with my phone. I was never really that much into phones to be honest. I reluctantly joined the mobile-toting brigade when I was travelling for work a lot and apparently I had to have one in case I was needed for something. It seemed to upset people that I rarely turned it on. As far as I was concerned, the phone was for my convenience, not theirs.

I only really started using a mobile phone when I got my first smartphone. Well, I say smart, it was I recall running some version of Windows. But it had a slide-out keyboard, and that made it infinitely more useful.

Grumpiness amplifier

These days I work from home. I don't need a phone, and even when I travel, I am more likely to be carrying a laptop or tablet, or both. They are just so much more useful (although they do make more of a racket if you forget to turn them off in the cinema). SMS messages are annoying (not to mention unreliable – do you know how they work? Don't bother finding out, it will just scare you). And if I really want to actually 'talk' to someone (which means it is a personal call) then I generally like to see them as well, which leaves the phone as the worst option. The things it does well are things I don't actually want to do. I sometimes check what the time is on it…

So, really, the relationship is over. I don't want a phone, I want a computer. I want to do the things a computer can do, I don't care about the things a phone can do. I think 'phones' should go the way of fax machines – only used by people you are better off avoiding. Someone should make a computer that is sort of phone-sized for portability, then I might be interested again. And could they do it soon? My contract is almost up…


Lots of books on random subjects – these cover the entire wall, and includes the obligatory collection of Linux Voices.
Eee PC – great robust PC for travelling.
Self-built RepRap 3D printer. Works great with Linux!
PCIe cards for which I maintain the kernel driver.
Speccy 128k.
Old Dell 9150 tower PC, with audio breakout panel

My Linux Setup Mark Einon

Software consultant, mainly with civil aviation software, part-time kernel hacker and kernel maintainer, tinkerer.

Q What version of Linux are you using now?

A Left to right – Debian Wheezy on the tower, along with many VMs – CentOS, Arch and whatever else I'm playing with. Wheezy on the Eee PC, Wheezy on the battered i7 laptop. To summarise: Debian Wheezy. I also have two RasPi's in a cupboard, one running an email server and the other *irrsi/tmux*, so I can keep up to date with the #LinuxVoice channel.

Oh, also The Kernel on the VIC-20, Kickstart on the Amiga A600 with a CF card hardware mod.

Q And what desktop are using on these machines?

A Xfce for less capable machines, otherwise the default Gnome. I'm not a big desktop user, preferring a terminal running tmux; with Mutt, Vim, Taskwarrior, SSH etc, the setup for which I keep on GitHub and pull down when needed to any machine.

Q What was the first Linux distribution you used?

A Fedora Core sometime around 2004, which I dumped when it got harder to install your own kernels and I swapped to Ubuntu. Ubuntu was always troublesome to do kernel development on due to their random frequent userspace changes, so I later moved to Debian which is lovely, transparent and stable, and have been using it ever since.

Q What Free Software/open source can't you live without?

A The bread-and-butter of the Linux kernel. Everything else on top is jam.

Q What do other people love but you can't get on with?

A This newfangled internet thing – particularly social media sites and their quest to monetise you. It's very useful but can also be a waste of time. I stopped using Twitter as it became more wasteful than useful. G+ is OK though.

If you support open source software, The Open Source Initiative needs your support.

I love open source & want to **help**

I support open source software and the OSI

I support the free & open source movement

Open is the better way

Because I believe in open source

Open Source is an idea that has changed the world for the better The OSI helps keep that idea alive

Wow. Been seeing a ton of tweets from @OpenSourceOrg the last few weeks Glad to see it's getting some energy back

I deeply believe in the OSI mission

OSI represents hope

I am a firm believer in what the OSI, and other open source communities, are doing to keep software transparent

I want to help the OSI because it is time to gave back after all the support they have provided

The OSI is redefining business

I have enjoyed the benefits of open source for many years and, now an open source contributor, would like to help support the open source infrastructure

Open source shapes our future

The open source movement needs this kind of **support**

The future depends on new ideas,alternatives and paradigms

The OSI promotes and protects **open source**

talented people to collaborate on finding solutions

Open source software also spurs innovation and collaboration allowing different people to experiment with their own ideas about how the software could work

Open Source Initiative provides a powerful **community** from across the globe

The **Open Source Initiative** (OSI) was founded on February 3rd, 1998, or "2/3/98" and in 2012, the OSI transitioned to a membership-driven organization, offering both Affiliate Memberships to non-profit open source projects and Individual Memberships to open source software developers, enthusiasts, supporters and end-users.

To celebrate both our founding and promote our commitment to a representative, community driven organization, the **Open Source Initiative** is launching our first ever

Individual Membership Drive

on our anniversary — February 3, 2015 — with a goal of 2,398 new Individual Members.

We invite you to join our Affiliate Membership, including Debian, Drupal, Eclipse, FreeBSD, Linux Foundation, Mozilla, Python, Wikimedia, Wordpress and many more, as well as our corporate sponsors, Google, HP, IBM, Linux Journal and Nginx to name a few, and of course the many other Individual Members from around the globe, who have already committed to promoting and protecting open source software development, projects and the communities that ensure its continued adoption and success.

Please join now at: www.opensource.org/join

The **Open Source Initiative** is a member-driven non-profit corporation with global scope formed to educate about and advocate for the benefits of open source development and to build bridges among different constituencies throughout the open source community. Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, greater reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. As a member of the Open Source Initiative you'll help develop and drive the direction of open source software, ensure the integrity of the Open Source Definition for the good of the community and protect open source licensing, creating a nexus of trust around which developers, users, corporations and governments can organize open source cooperation.