

PROUDLY INDEPENDENT SINCE 2013

LINUXVOICE

Virtualisation



Run a virtual system inside your Linux machine

September 2016 FREE SOFTWARE | FREE SPEECH www.linuxvoice.com



ULTIMATE SPEEDUPS

Get maximum performance from your Linux machine!

ARDUINO HARDWARE HACKING P28

PUBLISH A MASTERPIECE WITH CALIBRE P22

REVIEW: FEDORA 24 WORKSTATION P42

SPEED UP YOUR INTERFACE!

SPEED UP YOUR INTERWEB!

SPEED UP YOUR SERVER!

BURSTING WITH AWESOME TUTORIALS!

- ANSIBLE Harden many servers at once the lazy way
- APACHE Find out what's going on with log files and lots of graphs
- ELASTIC BEANSTALK Deploy a web app to Amazon Web Services

FREEDOM FOR FORMATS!
FRIDRICH STRBA

The Document Liberation Project – making sure you can read your own files.



ONE OF US! ONE OF US!
HELLO, LINUX

New to Linux? Start here – our introduction to the finest operating system in the world.



FREE SOFTWARE | FREE SPEECH

September 2016 £5.99 Printed in the UK



ISSN 2054-3778

RETROPIE > XATTRS > DIGIKAM 5 & MORE!

FOSSTALK LIVE 2016

A free evening of live Linux Podcasts
Saturday 6 August 2016

LINUX VOICE



Plus Stuart Langridge and Dave MegaSlippers

<http://www.fosstalk.com/tickets>

The Harrison, 28 Harrison Street, Kings Cross, London, WC1H 8JF
Doors 5pm

FULL SPEED AHEAD!

The September issue



BEN EVERARD

Long-term Linux user and best-selling author Ben is usually found knee-deep in either Python code or a tangle of wires.

When Les Pounder sent in the Arduino feature (page 28), he included the note, "I forgot how much I enjoyed hacking with Arduino." I played with one of my Arduinos this month, and had a great time as well. Personally, my enjoyment comes from the simplicity of the device: there's little extraneous hardware, and instead of an operating system, there's a bootloader. If you need any more features, you have to build them yourself. By being stripped bare, the Arduino forces you to think about exactly what you need your computer to do.

Don't worry, I'm still a Linux user at heart. This month I've been tuning my machines for maximum performance, and that means getting to know exactly how it's using the resources it has – it's a little like programming an Arduino.

Ben Everard
Editor, Linux Voice

THE LINUX VOICE TEAM

Editor Ben Everard
ben@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Editor in hiding Graham Morrison
graham@linuxvoice.com

Editor at large Mike Saunders
mike@linuxvoice.com

Games editor Michel Loubet-Jambert
michel@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors: Andrew Conway, Mark Crutch, Sebastian Göttschkes, Vincent Mealing, Simon Phipps, Les Pounder, Mayank Sharma, Amit Saha, Valentine Sinitsyn

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.

What's hot in LV#030



ANDREW GREGORY

This month's group test on desktop virtualisation has helped me turn my PC into hundreds of little PCs. Why run one distro when you can run all of them at the same time?

p50



GRAHAM MORRISON

I'm planning on writing a book on the finer points of Belgian beer making with the help of Andrew Conway's tutorial. Now I just need to finish a few more batches of golden brew to perfect the art.

p68



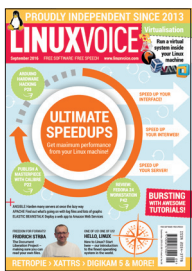
MIKE SAUNDERS

I'm planning the next step for my own operating system, MikeOS, where it takes over the world of web servers. I've been reading about log file analysis to see what the competitors can do.

p82

**SUBSCRIBE
ON PAGE 56**





Contents

If at first you don't succeed, Google the problem, then go back to using Debian.

Regulars

News 06
PS3 users in the US get a payout, there are new releases of KDE and Digikam, Linux desktop use passes 2%, and we're taking over Microsoft's Azure cloud platform too.

Distrohopper 08
Featuring one of our favourites at the moment – Manjaro 16.06 (aka the excellence of Arch without the hassle).

Speak your brains 10
What's going on at the heart of government (in software terms, that is), an indexing solution, love for Fedora and vote of thanks.

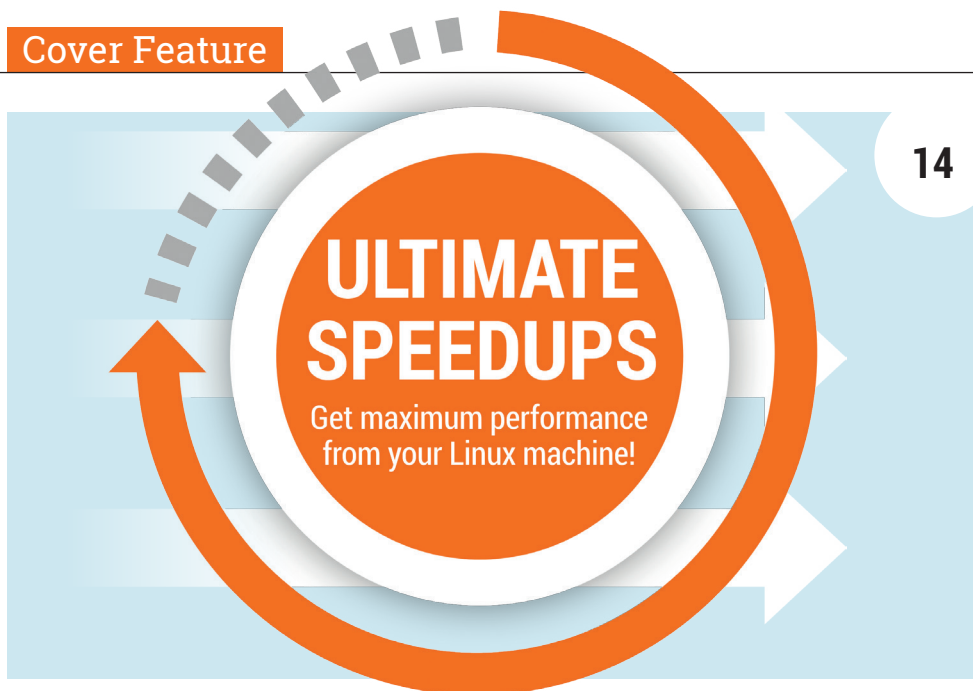
Subscribe! 12/56
Never again feel the gut-wrenching pain of missing out on your favourite Linux magazine. Subscribe today!

FOSSPicks 58
Free-range, grass-fed software, slaughtered at 18 months old and hung for 35 days, cooked on the bone and tasted thoroughly by Graham Morrison.

Core Tech 94
Files contain metadata including permissions – but what if you could add your own metadata to make files even more useful? With xattrs, you can!

Linux inside 98
The SpaceX Falcon 9 rocket – bringing the cost of space missions down with the help of the Linux kernel.

Cover Feature



Your bog-standard laptop can be a supercomputer if you just clean away the cruft and make a few simple tweaks – here's how...

Interview



Fridrich Strba

How the Document Liberation Project is working to free your data.

Feature



THE LINUX NEWBIE GUIDE

Welcome to Linux

What is this crazy little thing called Linux? It's only the safest, smartest and most secure way to run a computer!

FAQ

Flatpak
Distro-independent packaging has long been the dream for Linux – is Flatpak the chosen one?

Group Test

Virtualisation apps
Test new distros, write code in a clean environment or just mess about with a virtual machine.



SECRETS OF CALIBRE
TURN TO PAGE 26

SUBSCRIBE ON PAGE 56



Feature

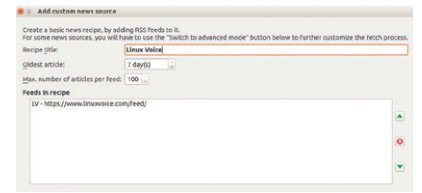
Tutorials

BUILT WITH

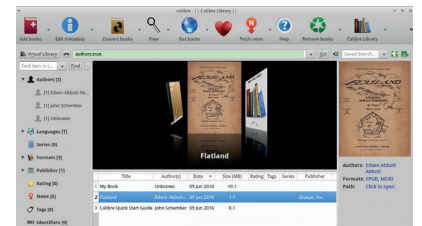
TM 28

ARDUINO

Build all the things!
How a humble microcontroller kick-started the march of the makers.



Calibre 66
Edit your own personal online newspaper to get the news you want without any of the rubbish *Game Of Thrones* memes.



Publish books 68
... and staying with Calibre, why not write your own fantasy epic with free software and publish it yourself?

Raspberry Pi 74
Use a barcode scanner and little bit of Python to control characters in *Minecraft*. it's like *Pokemon Go* but better!

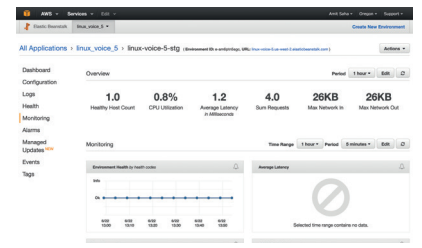
Ansible 78
Keep on top of server security, updates and configuration by controlling loads of machines all at once.

Coding

```

logins cat: access.log | awk '{print $8}' | sort | uniq -c
400 HTTP/1.1
4169 HTTP/1.1
4482 200
524 403
75 404
13 308
5 302
logins root: access.log | awk '{print $4}' | awk -F/ '{print $1}' | sort | uniq -c | d
11 20823
02 24562
03 24722
04 24651
05 22769
06 24255
07 28254
08 27316
09 28015
10 27231
    
```

Amazon Electric Beanstalk 82
Deploy a web application with Amazon's super-scalable hosting platform.



Apache log files 86
Find out what's going on with all those blinking lights and whirring hard drives.

Reviews

Fedora 24 42

Our verdict on one of the big beasts of the Linux desktop – the community-run derivative of Red Hat, by the people, for the people.

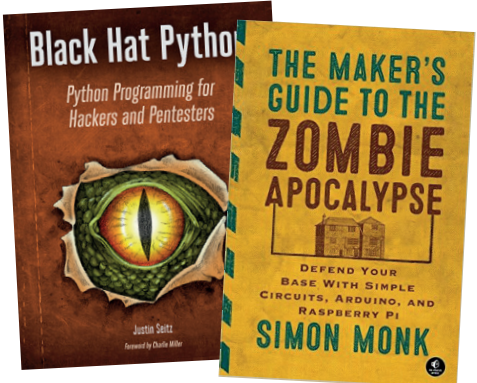
RetroPie 43
Play retro games on a humble Raspberry Pi with this fabulous emulation platform.

FritzBox 44
Replace your ISP's rubbish router with something that'll give you more control over your network.

Digikam 45
Photographers are spoiled for choice on Linux – here's one of our favourite apps for snappers.



Gaming on Linux 46
To prepare you for the book reviews page, here's a quick review of a zombie survival game – sadly, Arduinos are not included in the gameplay.



Books 48
When zombies attack and the world is in flames you'll be glad you learned how to make little Arduino- and Raspberry Pi-powered devices.

NEWS ANALYSIS

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

Opinion

Seperate yet united

Open Source developers aren't working for free – they're working for themselves.



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

Open Source is thus what happens when people choose to work together on the same codebase rather than working separately

So you're thinking of devising an open source strategy for your business. The most important concept to understand when devising an open source strategy is the separation of community interests and commercial interests. What does that mean?

Open source can be defined as the co-development of software by a community of people who choose to align a fragment of their self-interest in order to do so. The commons in which they work contains software free from usage restrictions with guaranteed freedoms to use, study, modify and distribute it – “free software” – because of an OSI-approved copyright licence (an “open source licence”). All such licences grant permission to anyone to use, improve and share the licensed source code and the object code that it produces. The licence creates a safe space for collaboration, where everyone has permission in advance to innovate however they want.

The community members each work at their own expense in order to achieve a shared outcome that benefits all, including themselves. When they create an enhancement, fix a defect, participate in a

design, they are not “working for free” or “donating their work” so much as they are “participating in co-development”.

Open source is thus what happens when several different people choose to work together on the same codebase rather than working separately, liberated to do so by the four freedoms that the licence protects. Each of them is there for their own reasons. There is no pooling of funds to pay for work to be done, because everyone is solely responsible for their own costs. The only funding pool a project needs is to cover the costs no-one should bear alone, such as project infrastructure and administration or for tendering for paid work that is in the interests of all.

It's not about the money

As a consequence of this connected-but-separate status, there is no fiscal power that any contributor holds over others – no-one has the right to tell the others what to do. There will sometimes be a non-profit organisation for administrative reasons, and often a technical co-ordinating group to make sure releases can be scheduled and duplication can be avoided. To maintain trust, enable development transparency and permit individual privacy, it's reasonable to devise and apply governance that asserts

community norms. But beyond those organisational essentials, an open source community is inherently neither a non-profit or a for-profit organisation; profit is an orthogonal concept.

Some of the contributors might be present for direct profit from the code, but the community as a whole is actually a mesh of different participants, all with their own motivational models and all paying their own way to achieve them outside the context of the community. Communities do not have business models. If the motivational model of some participants involves business, that harms no-one. But the community itself is about the liberty to align interests, not about the presence or absence of profit – that is purely the domain of the participants privately.

Thus in a healthy open source community, I'm free to maintain my privacy around my motivations and how I'm funding my involvement if I wish. On the other hand, I'm able to work in an environment of transparency where all the code is known, all its origins are known, all its defects are potentially known.

That combination of transparency with privacy is, in my opinion, a primary characteristic of an open source community. Communities without the seminal Apache Software Foundation rule “if it didn't happen as a matter of open record, it didn't happen” are closed, regardless of the software license. Open source is about transparency at the community level but also about the privacy of the individuals involved.

There is no fiscal power that any open source contributor holds over others – no-one has the right to tell the others what to do

CATCHUP

Summarised: the biggest news stories from the last month

1

KDE Plasma 5.7 released

KDE Plasma 5.7 was released on 5 July 2016, and brings about extended Jump List Actions in KRunner for quicker access to certain tasks within an application. Also, the Agenda view in the Calendar is back, while many improvements have been made to the Volume Control applet (such as the ability to control volume on a per-application basis).

Accessibility is better in the new version as well: Breeze icons within applications are now tinted depending on the colour scheme.

2

So long, 32-bit distros...

If you're running a PC or laptop bought within the last five years, chances are that it has a 64-bit chip inside. Now some distro makers, such as Ubuntu and OpenSUSE, are considering dropping support for 32-bit processors. As OpenSUSE Chairman Richard Brown says: "32-bit support doubles our testing burden (actually, more so – do you know how hard it is to find 32-bit hardware these days?)". Of course, some smaller distros will continue to support the older hardware.

3

Linux Mint 18 "Sarah" released

Based on Ubuntu 16.04, Linux Mint 18 is a long term support (LTS) release that will be supported until 2021. The Mate edition includes version 1.14 of the Mate desktop, which sports better GTK 3 support and better management of Python Caja extensions.



4

Linux finally reaches over 2% of desktop market share

We've all dreamt about the "year of Linux on the desktop" for as long as we can remember, but things are starting to go in the right direction at last. As of early July 2016, according to W3Counter stats, Linux now accounts for 2.8% of computer users accessing the web. It may not sound like much, but it's still way better than the ~1% around which it was hovering for many years – fingers crossed there's more success to come.

5

Slackware 14.2 released

We like Slackware. It's very Unix-like at its heart, eschews complicated package management systems for a simple tarball-based approach, and provides an alternative init and process management system to Systemd (not that we hate Systemd, but we appreciate diversity). This new release includes 4.4.14 and GCC 5.3, along with Perl 5.22.2, Python 2.7.11, Ruby 2.2.5, Subversion 1.9.4, Git 2.9.0 and KDE 4.14.21. To download it or buy a boxed set and support the project, see www.slackware.com.

6

Sony pays out to angry PS3 owners

At launch, the PlayStation 3 let you install Linux, which was a selling point for a small but vocal number of users. Ridiculously, Sony later issued a firmware update removing this feature – and you needed to install this update to play newer games.

Some Linux fans took Sony to court because of this move, and now it looks like Sony will pay out \$55 to each user who had Linux installed and then couldn't use it. Not much then, but at least it's something.

7

Digikam 5.0.0 has been released

It has been two years in the making, and a huge amount of work, but Digikam 5.0.0 is finally here. Most notably, this photo management tool has been updated to Qt 5, with all Qt 4 and KDE 4-related code removed. In addition, the app uses more Qt libraries and fewer KDE ones, making it easier to port to other OSes. Digikam 5.0.0 is also considerably faster than previous releases, thanks to optimisation work done by a Google Summer of Code-sponsored developer.



8

A third of Azure VMs are now running Linux

Microsoft and Linux really are making for interesting bedfellows these days. Previously, only a quarter of virtual machines on Microsoft's Azure cloud infrastructure were running Linux, but now that has gone up to a third. Cynics would say that Microsoft is simply going where the money is, and Linux is huge in the cloud, but we're happy to see the company being a bit more open to other platforms now. It's a lot better than the terrible Steve Ballmer days. Sunlit uplands are ahead!

DISTROHOPPER

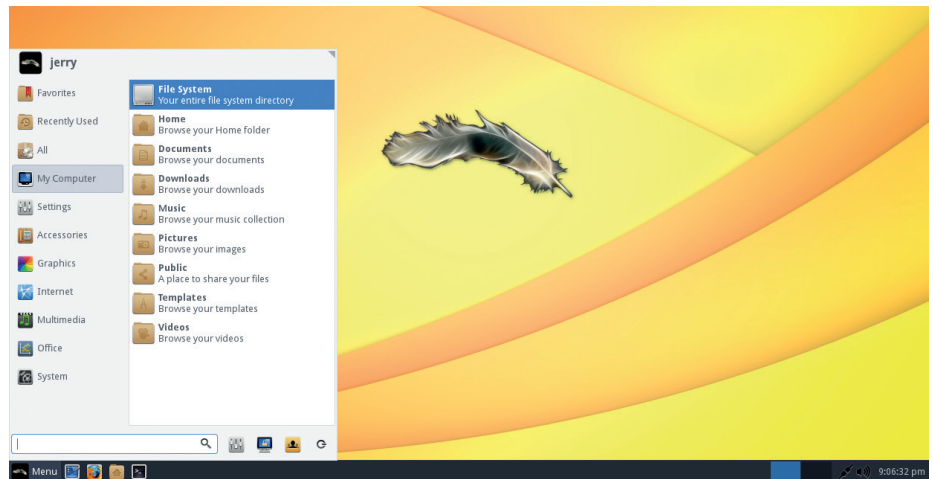
What's hot and happening in the world of Linux distros (and BSD!).

Linux Lite 3.0

No fuss, no nonsense.

This no-nonsense distro has recently released version 3.0, based on the Ubuntu 16.04 LTS codebase for added stability. There's been some considerable updates since 2.8, including new themes, a new login manager and an overhaul of the very sleek and easy-to-use "Lite Software" software installation manager, along with the 16.04 package updates. While it clearly brands itself as a light distribution and does earn its name in this regard, it is perhaps better to look at Linux Lite as an extremely complete desktop environment, just without all the bulky and useless bits. This is made immediately evident through the use of Xfce, which takes a similar approach in trimming things down without having a negative effect on usability and appearance.

While Linux Lite does run on very modest hardware, the minimum being a 700MHz processor and 512MB of RAM, it is also well suited to those with high-end hardware who simply want to use the extra horsepower for things other than desktop effects. Linux Lite



Linux Lite's approach seems to have struck a chord, as it is growing in popularity.

thus has an appeal that most lightweight distros lack, offering no bloatware without sacrificing functionality.

The distro is also extremely easy to set up and to maintain, with the ability to get going in minutes and things like updates, software installation, version upgrades and driver installations done through a simple unified GUI that entry-level users would have no

issue with. As such, this fits in nicely with the idea of a system for people who just want to get things done, with the distro staying well out of the way, not being demanding in terms of configuration or resources. While Linux Lite doesn't break the mould in any meaningful way, it's hard to think of a better alternative for those looking for a solid workhorse distro.

Manjaro 16.06

Arch for human beings.

Manjaro has, for a long time, been the distro of choice for those looking to delve into the world of Arch Linux without feeling overwhelmed. This latest stable version of Manjaro, dubbed "Daniella", ships with Xfce 4.12, and while this desktop environment remains the default choice for this distribution, there is also the KDE Plasma flavour, as well as the Net edition, which comes with no preinstalled desktop environment.

The biggest changes from 15.12 include an update to kernel 4.4 LTS, updates to the

Pacman package manager, a new "Manjaro Settings Manager" to install and remove different kernels should 4.4 not suffice and, for the KDE version, a KCM module to integrate the settings manager into Plasma 5's native settings, as well as updating Plasma itself to version 5.6.

It remains an extremely attractive distro, not only for its rolling release model (for those who prefer it) and "Arch made easy" philosophy, but also for a myriad of other reasons such as its easy installation, user-friendliness and "out of the box" usage



The Manjaro developers have described the Xfce flavour as the "flagship" edition.

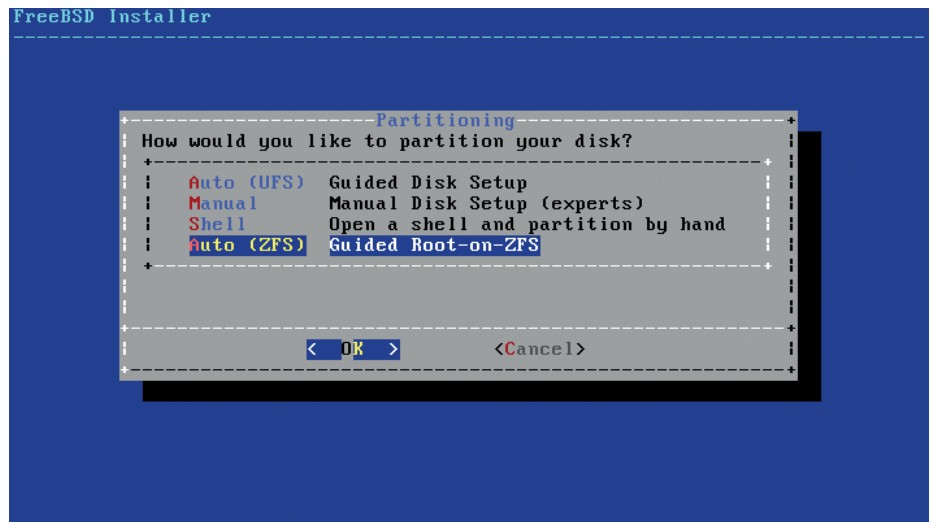
once installed. While maintaining Manjaro requires a bit more knowledge than using, say, Ubuntu, and wouldn't be worth recommending to someone taking their first steps into Linux, non-technical users could use it without issues.

News from the *BSD camps

What's going on in the world of FreeBSD, NetBSD and OpenBSD.

NetBSD is celebrating the 50th release of the *pkgsrc* package management system with a series of developer interviews to mark the event, which can be found on the project blog: blog.netbsd.org/tnf. It was initially released in 1997 when it was forked from the FreeBSD Ports collection, and both of these remain the default package management systems on each operating system. At the time of writing, *pkgsrc* contains over 17,000 packages and is available on multiple systems. NetBSD itself saw the first bugfix release of the 7.x series in the form of version 7.0.1. Meanwhile, the system's firewall software *pfSense* has seen a major point release in the 2.3 series, incorporating over 100 changes.

Meanwhile, FreeBSD has now joined Linux as the latest operating system to be included alongside Windows in Microsoft's Azure cloud computing service. While previously customers could upload custom images, FreeBSD 10.3 is now one of the available images pre-built by Microsoft. While Amazon's competing Web Services



A ZFS partitioning option on the FreeBSD 11 installer, alongside other options.

has offered BSD (and Linux) for a long time now, this step shows Microsoft recognising that, in terms of market share, Windows simply can't compete with Unix-like operating systems in these applications. The company, which has been making a series of similar moves in recent years, will

be supporting the BSD clients directly so as to "remove the burden" from the FreeBSD Foundation, to which it will be sending code.

Also within the FreeBSD world, the ZFS Fault Management Daemon has been integrated into FreeBSD 11, adding increased functionality for the filesystem in what seems to be a growing trend towards its support and adoption. On the DragonFly BSD fork, its kernel now supports NVMe (Non-Volatile Memory Express), meaning that users can now make use of this modern storage standard.

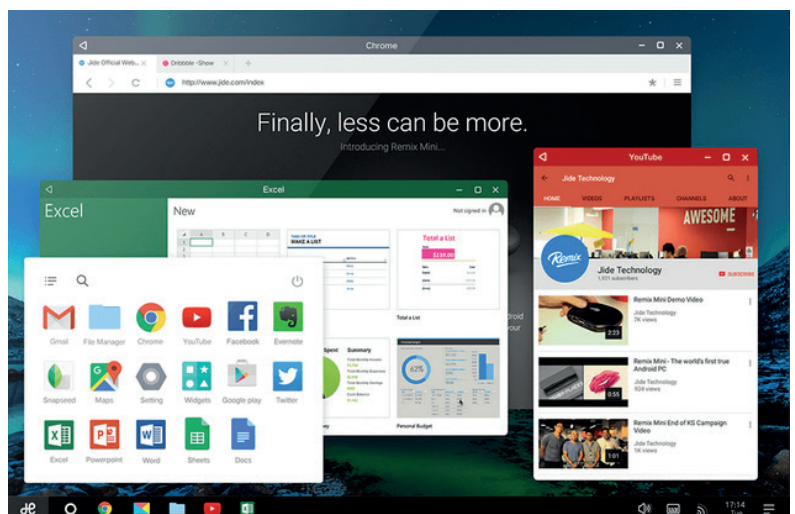
FreeBSD has joined Linux as the latest OS to be included alongside Windows in Microsoft's Azure cloud computing service

Remix OS for PC

Linux has now dominated pretty much every other space out there, but still only manages third place on the desktop. So how about reversing that by getting a mobile distribution that almost everyone is familiar with and bringing it to PCs? Remix OS does just that, by bringing Android to the desktop.

Remix's performance on x86 leaves ARM devices in the dust, and the system itself is pretty seamless, running Android applications on the desktop. While using things like office applications is far more productive than on a touchscreen, seasoned Linux users may find it less convoluted to install a lightweight Linux distro with a vast array of available software specifically designed for the desktop.

Though Remix OS is unlikely to be the one to achieve mass-market desktop Linux glory, as it's becoming more likely that Chrome OS and Android are to become one and the same in the future, either case would be a bit of a hollow victory. While technically, the Linux kernel would be running on a lot of machines if such a thing were to gain traction, it would be at the cost of many of the values and freedoms associated with GNU/Linux. Remix OS itself is free but not libre, while the Google experience would also be somewhat of a walled garden.



Remix OS can run native Android applications on the desktop.

YOUR LETTERS

Got an idea for the magazine? Or a great discovery? Email us: letters@linuxvoice.com



STAR
LETTER

LINUX.GOV

Again a great magazine!

Some time ago I heard about Collabora doing a government office suite. Is that live now? What about the ODF requirement?

www.collaboraoffice.com/solutions/collabora-govoffice
Richmond Makerlabs, Ham United Group

Andrew says: Collabora did indeed sign an agreement with the UK government in October 2015 to provide office software, but we haven't

heard anything from them since. Rather than speculate here, I think the best thing to do is to get in touch with our chums at Collabora and find out from them what's going on. One thing that we can be sure of, however, is that progress will be slow: Free Software implies a cultural shift that's at least as important as any financial saving, and as the Cabinet Office has now started using Google Docs (!) it looks like we won't be 'taking back control' any time soon.



Free software for the public sector makes so much sense; it's a shame it has to happen so slowly.

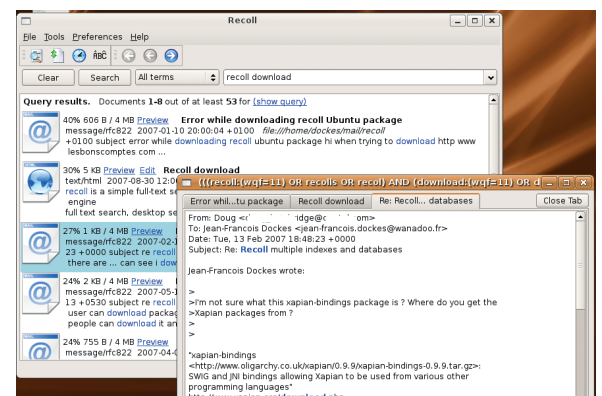
#INCLUDE LV_FULSOME_PRAISE

In his letter, Ken Riley asked for an index of Linux Voice articles. *Recoll* works very well on a downloaded collection of .pdf files, including other formats. *Recoll* builds a full-text index that you can query; it includes word proximity and stemming, and has an easy-to-use GUI. You can set *Recoll* for periodic automatic updates of the index. *Recoll* is available in the Debian repository and it has good help documentation.

Though good, *Recoll* can't overcome typos and dodgy copy editing.

Regards, Andrew Shead

Andrew says: Thanks Andrew. It sounds like this might make a decent stopgap solution, which, to be honest, is what we're all about.



Sadly, *Recoll* is not a platform puzzle game set on Mars. But it can help you make an index out of PDF files.

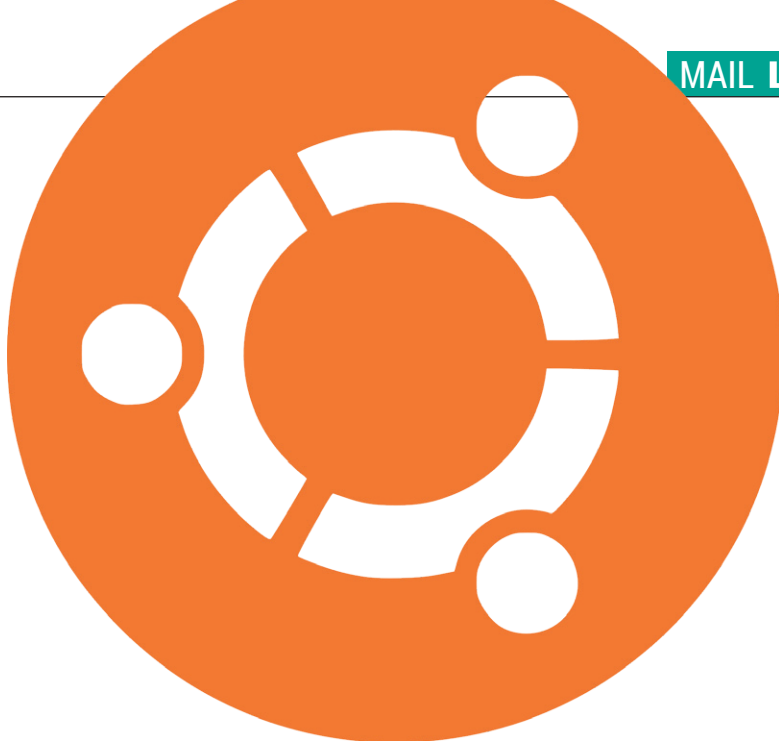
SO LAST YEAR

We get it: you guys like Ubuntu. It's the best thing since sliced bread. But the world has moved on. Unity is broken, and one of the biggest features of the last release was that it doesn't spy on you anymore! As if we should even need to be told that!

Fedora is and always has been the One True Way. It's a proper desktop distro for people who want to get things done (sorry Arch users, but most of us have better things to do than read the wiki before every piffling little update). It respects your freedom, and it Just Works. Carry on.

Iain McAllister

Andrew says: It chills me to the bone when Graham goes offline when he's updating Arch; we just don't know when we'll ever see him again, or what mental state he'll be in when he does resurface. Ubuntu has never done that to me, and if it did, I know that a quick Google search would bring up the answers to any problems instantly. It's worth looking into Fedora for the benefit of



those readers who haven't tried it in a while, for its excellent community as much as anything (which incidentally is by far the best feature of Ubuntu). Watch this space for more Fedora action!

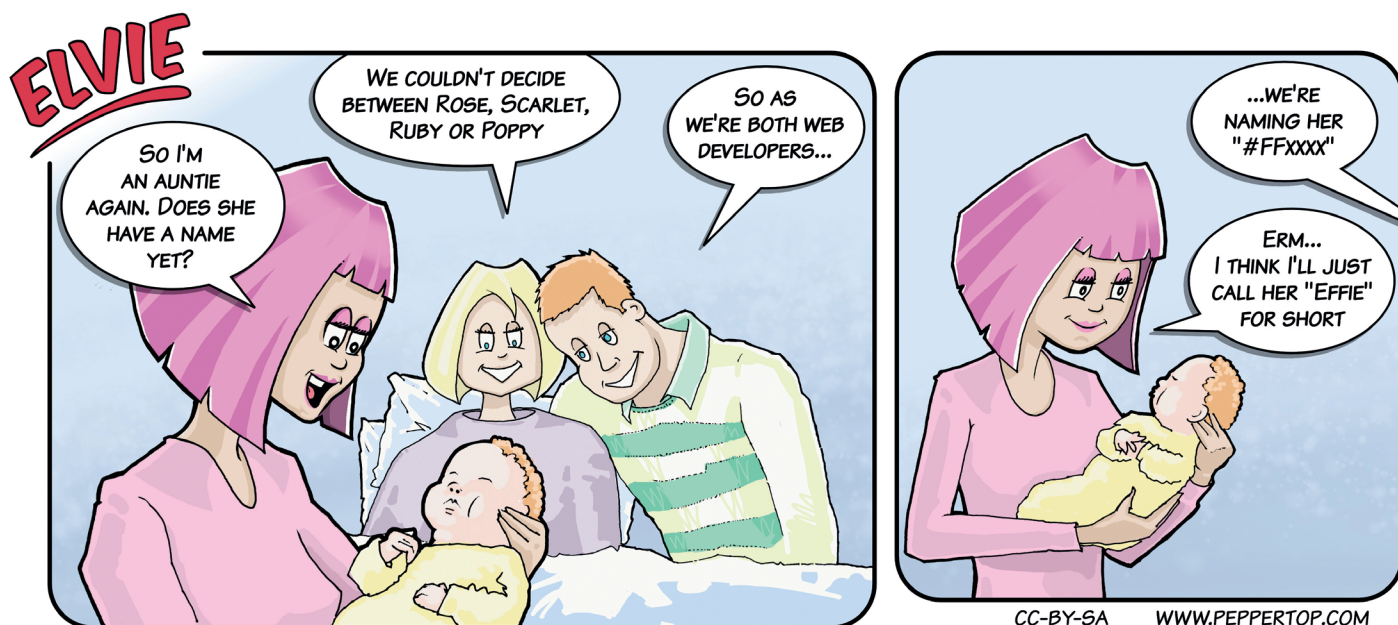
Remember kids: spying is bad. Unless you're the Home Secretary.

FREEDOM ISN'T FREE

Thanks for the interview with Jim Killock; I didn't agree with everything he said, but it was nice to get a few proper, thought-out reasons why surveillance is bad. My own thoughts are that it wouldn't matter a damn whether the security services can read my email if I get blown up by a terrorist, and I'm really not interesting enough to spy on, but I do appreciate seeing the counter argument presented in a sensible manner. I haven't changed my mind, but I have at least thought about the implications, and that's to your and his credit.

James Olssen, Michigan

Ben says: Thanks James. In these bonkers, vitriolic times, it's reassuring that we can all get along despite our differences of opinion. Every time someone earnestly compares the UK with North Korea I sigh inwardly, because it's just not that simple. The Open Rights Group is making the case in a proper, nuanced manner, and it deserves our support. 🐧



Subscribe

shop.linuxvoice.com



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

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

US/Canada subs prices

1-year print & digital: **£95**
12-month digital only: **£38**

Get many 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

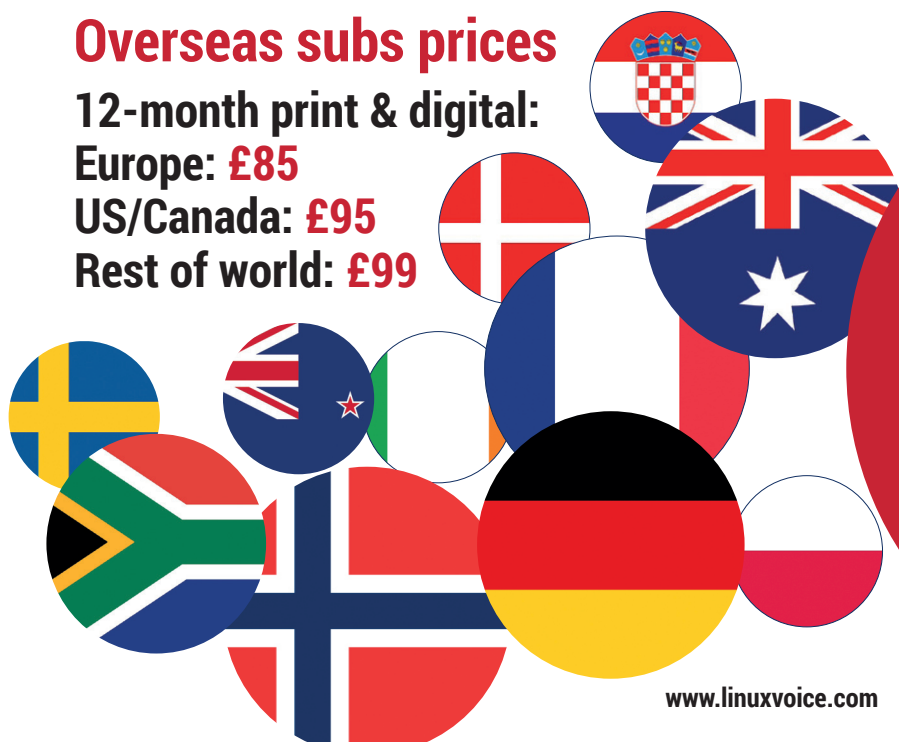
Payment is in Pounds Sterling. 12-month subscribers will receive 12 issues of Linux Voice a year. 7-month subscribers will receive 7 issue of Linux Voice. If you are dissatisfied in any way you can write to us to cancel your subscription at subscriptions@linuxvoice.com and we will refund you for all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

ULTIMATE SPEEDUPS

Get maximum performance from your Linux machine!

Having a fast machine isn't all about spending money on the latest hardware: it's about tuning the software to get the most out of the hardware you've got, whether that's a Raspberry Pi or a high-powered server. Even the most humble of modern computers is vastly more powerful than the machines of a decade ago, and we could quite happily run Linux at the turn of the century, so if things are starting to run like a tortoise in treacle, it can't be the hardware to blame.

Just like all machines, computers need a tune-up every now and again for them to perform at their best. There's no set list of things that you should do to make your computer run faster – instead, there are things you can investigate, and trade-offs you can make to balance performance with user experience.

Let's banish waiting time to the annals of computing history and make our computers as fast as they can be, with a new, leaner, faster environment. Strap yourself in!

Just like all machines, computers need a tune-up every now and then for them to perform at their best

SPEED TEST YOUR MACHINE

If you don't know how fast you're going, you won't know if you're going faster.

The first step in trying to increase the speed of your system is to find out how fast it's currently running. Once you start monitoring the speed, you can see what the effects of a particular change are, and you can roll back anything that makes things slower.

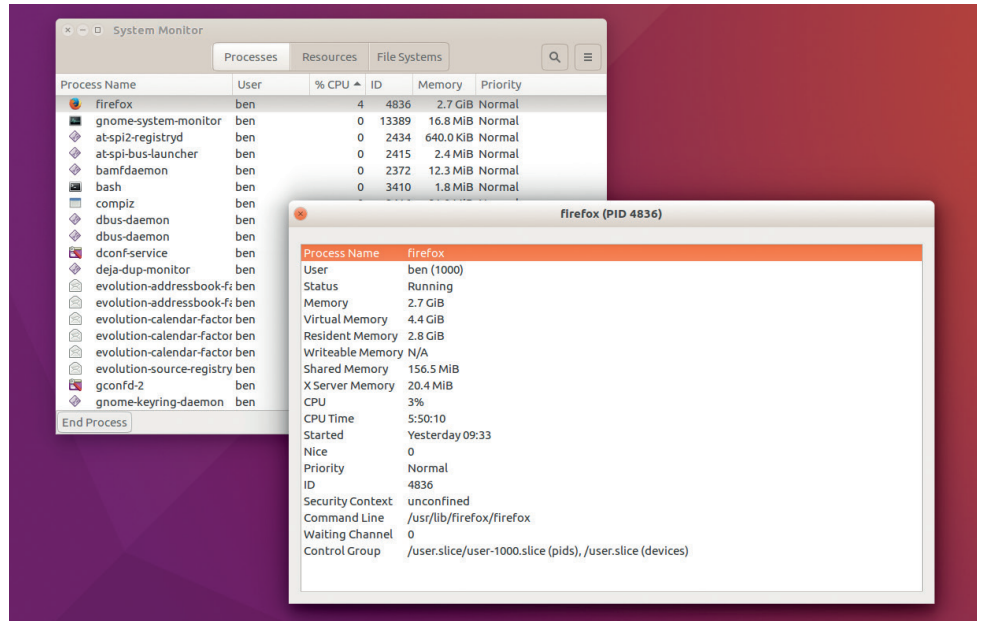
There are two basic ways of testing a computer's speed: benchmarking and performance monitoring. Benchmarking is where you run a repeatable test under different setups and see how the computer performed; while performance monitoring is seeing how your computer is performing as you're using it normally. The advantage of benchmarking is that it is a perfectly fair test, because exactly the same thing is run every time. The advantage of performance monitoring is that the data is more relevant because it's running the exact tasks that you run normally.

The simplest benchmarking tool is *GNU Time*, which measures how long other commands take to execute. For example, you can use it to test how long it takes to unzip a file (which tests processor and disk performance) with:

```
time tar xvf myfile.tar.gz
```

Using **time**, you can benchmark specific tasks that you do regularly rather than relying on general benchmarks.

If you want to really push benchmarking as far as you can, the best tool for Linux is the Phoronix Test Suite (PTS). This is a framework for running benchmarks and comparing



Gnome-system-monitor lets you see what's going on with a clickable user interface.

the performance across different systems or setups. There are a huge number of benchmarks available, and you can easily spend more time benchmarking your system than you

In Ubuntu, the configuration file is **/etc/default/sysstat**. Open this file and change the line:
ENABLED="false"
to

The best benchmarking tool for Linux by far is the Phoronix Test Suite (PTS)

could possibly save by increasing your performance, but if you're a performance geek, there's really no other tool that has the breadth of benchmarks that PTS has.

Real-world performance

Our go-to tool for performance monitoring is *Sysstat* (available in almost all distros' repositories). *Sysstat* collects and stores statistics, so you can go back and look at past performance issues. Since *Sysstat* is a monitoring tool rather than a benchmarking tool, it can take more work to understand the output. However, it can potentially tell you a lot about the real-world performance of your machines. Once you've installed *Sysstat*, you need to make sure it's running properly.

```
ENABLED="true"
```

Then restart the service with:

```
sudo service sysstat restart
```

In the normal setup, *Sysstat* will keep logs of the last 28 days in your log directory (usually **/var/log/sysstat**). These are binary logs, so you can't view them using the usual commandline tools (such as **grep** and **sed**), instead they have to be interpreted by software that understands them. The most usual tool for the job is **sar**.

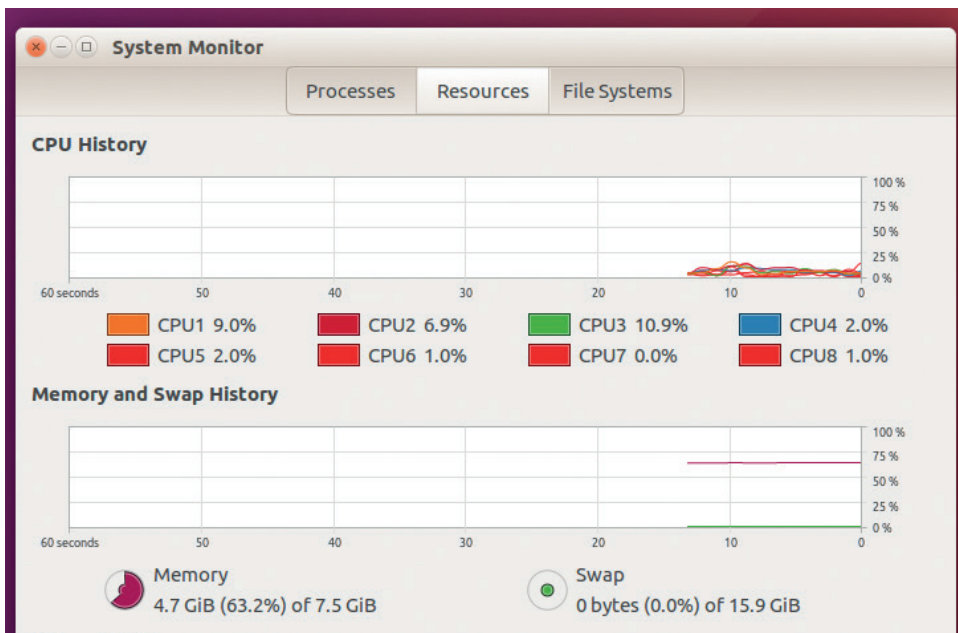
If you just enter **sar** at the command line, you'll see the CPU usage for the current day broken down into 10-minute averages, but the tool enables you to drill down much further than this.

You can get more up-to-the-minute data with the command:

```
sar 10
```

**Quick Speedup:
Boot faster**

Do you waste valuable seconds every time you start your computer with the boot menu? You can reduce the amount of time grub waits before starting by editing the **/etc/default/grub** file. Change the **GRUB_TIMEOUT** option to something smaller such as 2 (seconds). You'll need to save this file and run **update-grub** for the option to be picked up.



You can see how your resources are being used over time with `gnome-system-monitor`.

Quick fingers
 A significant amount of time, the thing slowing down the computer is the person sitting in front of it. In general, the more you use the mouse, the slower it is to perform actions, so learning keyboard shortcuts for your desktop and most commonly used programs can give you a significant speedup.

which will continuously output the data every 10 seconds.

The default `sar` output will aggregate the usage of all CPU cores, which gives a good overview of the system, but it can hide some problems. For example, if you have a single-threaded application maxing out one CPU core of a quad-core system, it would appear as though the CPU was 75% idle, but your computer performance would still be bound by the CPU. You can view the statistics for all cores separately

sar -P ALL 10

Keeping an eye on CPU usage will give you an idea of how much of your CPU time you're using. If you get close to 100% usage on one or all of the cores when your machine is running slowly, then you need to focus on reducing the CPU load in order to speed

up your machine. However, you may find that you still have CPU cycles to spare when running slowly because there are many things that can affect the speed of a computer.

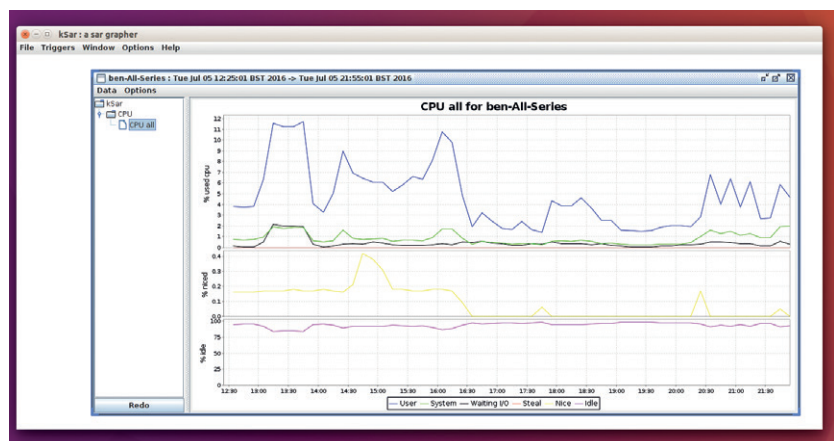
Not just the CPU

Your computer has RAM for storing the data and code of running programs – this is the fastest data store on your machine. It also has a hard drive to store data that's not currently in use, which is much larger than RAM but also much slower. The link between these is known as swap – this is a bit of the hard drive that acts like RAM. If you have too much data in RAM, your machine will put some of the contents of RAM in swap. This enables you to run more software than you can fit in

RAM, but it also slows down the machine significantly. If you've ever experienced a machine running very slowly as you switch from one application to another, this is likely due to the process of changing the bits of data between RAM and swap. In performance monitoring, it's important to look at both memory utilisation and swap usage.

Passing the `-r` flag to `sar` will output memory statistics. The most important column here is the `%memused`, which will tell you how much of the memory is currently in use. You can see the swap statistics with the `-S` flag. Using swap isn't inherently bad, but if your system's running slowly and you have consistently high amounts of swap usage, you should consider using applications that use less memory, or look at upgrading the amount of memory you have.

Monitoring your system enables you to see how it's performing without having to push it to its limits (as a benchmark would). For example, you might have a routine task that you run periodically; by monitoring the system resources during its execution, you can tell if it's using more or less resources than usual. You can also use it to see exactly which resources are being stretched when a system is reaching its limits. Depending on your setup, the limiting factor could be the CPU, the network port, the amount of memory or the disk space. By monitoring the resources used, you can find the area that's limiting performance, and after you've discovered where the problem lies, you can take steps to avoid the problem in the future.



Having trouble understanding the output of `Sar`? You can use `Ksar` to produce graphs showing how the usage changes over time.

SUPERCHARGE YOUR DESKTOP

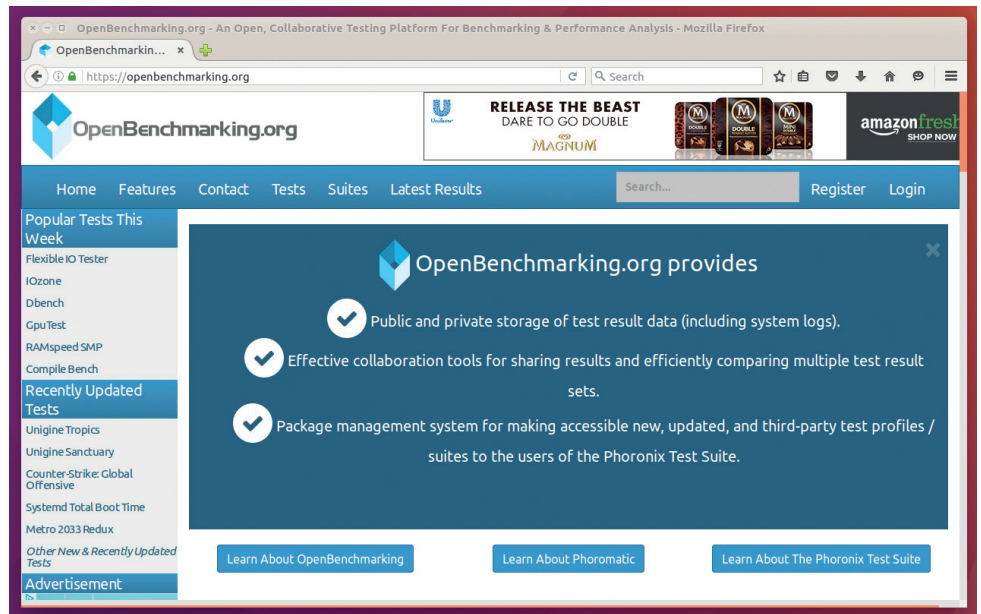
Get your Linux PC running at full throttle.

By far the biggest thing you can do to speed up a desktop Linux computer is to select an appropriate desktop environment. The two key factors to consider are the amount of CPU time a desktop uses and the amount of memory it uses. You can see how much of a problem these are for you by monitoring your machine's resources.

Memory usage of the desktop will be less of a problem when you first start the machine, but as you open more applications (particularly memory-hungry software such as office suites or web browsers), you may notice that the machine slows down as it has to shuffle data in and out of swap space.

In general, if you want ultimate performance, or if you're using a slow machine, you should avoid the big three desktops – KDE, Gnome and Unity – as they're all designed with a lot of features that help the user, but which also use valuable system resources. Instead, try a lighter desktop, such as LXQt, Openbox, Moksha or Mate, all of which offer a functional, working environment without wasting your PC's resources on fluff.

The first simple test we do to check the speed of a desktop is a timed exercise of going from the login manager to opening the file manager to opening a specific file in a text editor. This simple benchmark will test the snappiness of the desktop because the faster the desktop performs, the quicker you'll be able to perform all these tasks. It requires a few different



If you use the Phoronix Test Suite, you can compare your machines performance to thousands of others on openbenchmarking.org

parts to work quickly including logging in and starting the file manager. There

disk caches. This memory isn't really free because it's in use, but it isn't

If you want performance you should avoid using the big three – KDE, Gnome and Unity

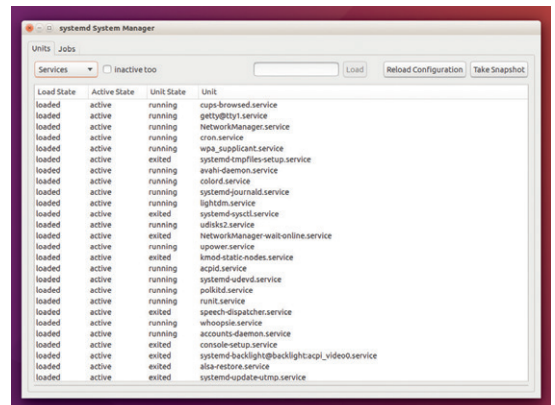
isn't an easy way to automate this test, so you'll need to use a stopwatch, and the performance will differ hugely depending on your machine.

A second, slightly more scientific test is to check the amount of memory used after first booting the system. You have to check this as soon as you've booted and logged in, because otherwise the software you're running (other than the desktop environment) will affect the figures. To check the amount of free memory, you can use

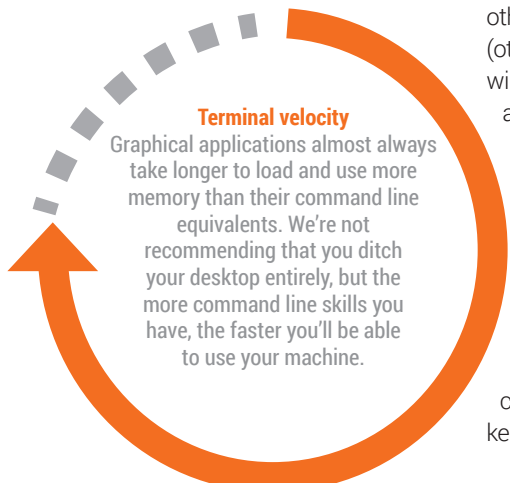
sar or open a terminal and enter **free**. The value you're looking for is in the **mem** line under the **used** column. The lower this is, the less memory your desktop is using, so the more memory there is available for other processes.

The reporting of memory on Linux can be a little confusing, because the kernel will use any spare memory for

really in use because it can be emptied instantly and nothing is lost (the caches can be repopulated from disk when they're needed). Some applications that output memory statistics include the caches in the used memory values (such as the **top** command) while



Systemd-ui gives you a graphical front-end to services (and other init-related options) for distros using Systemd.



others don't. If you're unsure, always check the output of the **free** command, as this gives a detailed breakdown including used memory and caches.

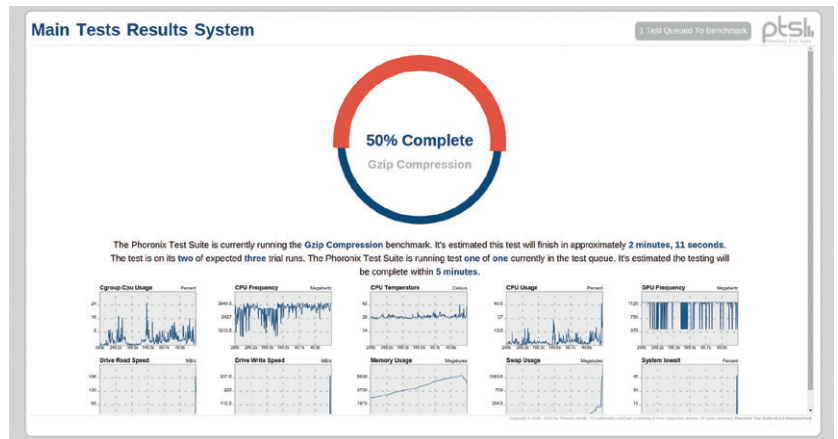
Beyond the desktop

The speed of a desktop is about far more than just the environment itself. The selection of all the different bits of software can have a huge impact of the speed of your system.

For example, *LibreOffice* is a hugely powerful piece of software, but do you need the full power? Editing the document for this article in *LibreOffice* uses 1.8GB of memory. In *AbiWord* it uses 1GB. In *Gedit* it uses 600K and in *Nano* it needs just 30K. Of course,

Parallel lives

There are some alternatives to common software that utilise multi-core machines better than the traditional tools. For example, *pigz* is a parallel implementation of *gzip* that's faster at compressing files on multi-core machines. You can also use *Gnu Parallel* to split a single command over more than one core.



Plunge into the world of benchmarking and discover the exact speed your machine runs at with the Phoronix Test Suite.

not all the pieces of software have the same functionality, so as with the desktop environment, the task for the user is to pick the best application for the purpose taking into account the amount of memory the software needs.

The process for checking the amount of memory a process is using is a little convoluted. First you need to find out the process ID of the application. You do this by opening a terminal and running :

```
ps -x | grep <application name>
```

Here, **<application name>** needs to be the command used to launch the software. This is often the software name in lower case, but for *LibreOffice*

it's **soffice** (a hangover from the previous name of *StarOffice*).

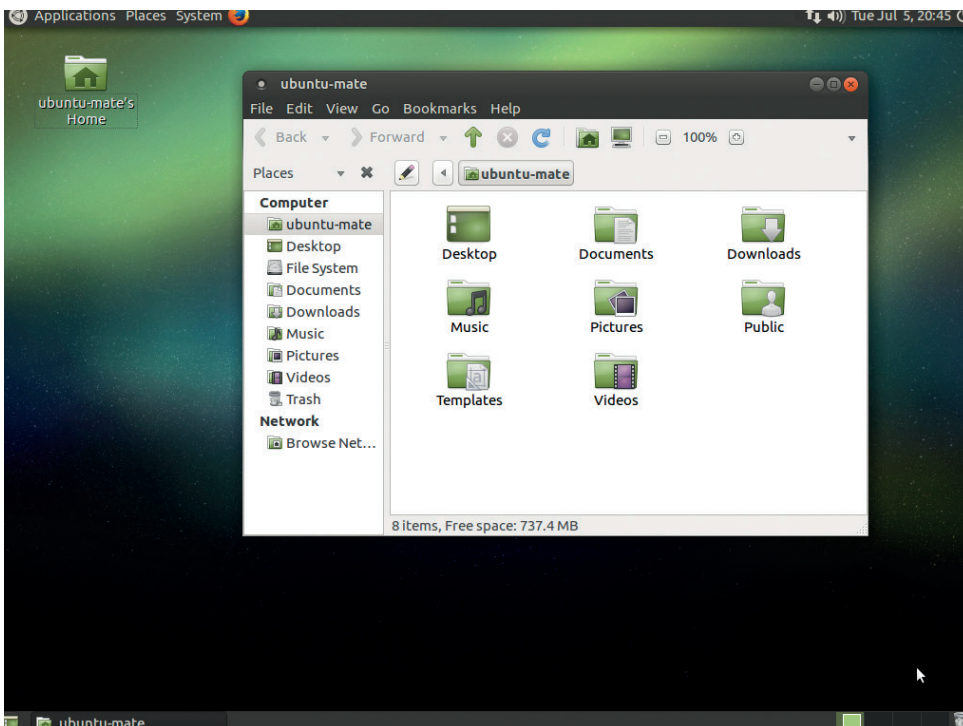
This will output a line for every running process that matches that line, one of which will be the **grep** command you used to search for the process, so you need to pick the right one. The first column on the line for the process will be the numeric process ID (or PID). You can then use the **top** command to find out more information about this process with:

```
top -p <process ID>
```

This will display various details about the specified process, including the current CPU usage. The three columns of data that are most interesting to us are **VIRT** (total amount of virtual memory used, including swap), **RES** (amount of memory currently in RAM but not including swap) and **SHR** (total amount of shared memory used).

This final column brings us onto another factor to consider when picking desktop software: the amount of shared memory they use. This basically comes down to the number of libraries an application uses, and the important thing to consider here is that libraries can be shared among different applications yet only take up a single slot of memory.

In desktop Linux terms, this means that if you use a *Qt*-based desktop (such as KDE or LXQt), all the *Qt* software can use the libraries already loaded by the desktop environment, but *GTK* software will have to load the libraries into their own chunk of memory. In other words, if you're short on memory, there is an advantage to sticking with software



Looking for a desktop that's fairly lightweight, but also stylish and easy to use? We recommend Mate.

designed specifically for your desktop environment.

When it comes to memory, the big thing that slows down your machine is shuffling data between RAM and swap. The Linux kernel decides what to transfer from RAM to swap by looking at the amount of free RAM, how recently a bit of memory was last used, and the swappiness setting.

Tuning swap

This last aspect – the swappiness – can be changed to make your system more or less ready to use swap. The benefit of having a high amount of swappiness is that, with more of the data in swap, there's more empty space in RAM that can be used quickly if needed. Any spare RAM is used by the disk cache, so a high swappiness can lead to faster disk access. The benefit of low swappiness is that data isn't put in swap so readily so your system can be more responsive when switching programs.

There isn't a definite answer for what the best swappiness value is, and the results of different values can vary depending on the amount of RAM on the system and the speed of your hard drive.

You can see your current swappiness setting with:

```
cat /proc/sys/vm/swappiness
```

By default, this is 60 on Ubuntu and many other distros.

You can change the swappiness by opening the file `/etc/sysctl.conf` as root (such as with `sudo nano /etc/sysctl.conf`). The swappiness is set on the line:

```
vm.swappiness = 60
```

Change 60 to any value between 1 and 100. In general, lower values are likely to work better on systems with

Add more power

If you've got a large command that you need to run on a low power computer, but you also have more computers available, you can share the processing load among them. For example, *distcc* shares the work of compiling software between many computers, and *Gnu Parallel* can split a shell script between many machines at once.



When it comes to eye candy without overly taxing your machine, the Moksha desktop is hard to beat.

more RAM and higher values better on systems with less, but the performance implications are complicated and will depend on exactly how you use your machine, so the best advice we have is to try changing it and see what happens.

At your service

Your Linux system has many pieces of software running in the background quietly getting on with their jobs. These services are started automatically when you boot your computer and usually keep running silently until you

running and which aren't. In general, it's a bad idea to stop a service unless you are sure that it's not needed, because some of them may provide behind-the-scenes functionality to other software. However, you may find software that you installed for a project that you no longer need. You can stop a service from running with:

```
service <name> stop
```

Doing this will only stop it in the current session. If you restart your machine, the service will restart. You can stop services from starting with:

```
service <name> disable
```

Your Linux system has many pieces of software running quietly in the background

turn it off. Some of the services you have running will be important, but some may just be wasting resources providing functionality that you don't need. The method for controlling services varies between distributions, but on most modern distros, you can see what's running with:

```
service --status-all
```

If that doesn't work, you should consult your distro's documentation for information on running services.

The output will depend on the distro you're running, but it should make it clear exactly what services are

You can reverse this with `service <name> start`, and `service <name> enable` if you decide you need the services in the future.

Most services are quite efficient if they're not under any heavy load, so disabling an unused web server isn't going to transform your machine, but it should speed up your boot times and reduce the amount of memory used when running. As well as the performance increase, disabling unused services will also improve the security of your machine, since there will be less running software for an attacker to probe.

SYSTEM CALLS

Upgrade your server performance without upgrading your hardware



```

ben@ben-All-Series: ~
┌───┴───┐
1  [  ] 3.3% 5  [  ] 1.3%
2  [  ] 3.4% 6  [  ] 1.3%
3  [  ] 7.2% 7  [  ] 10.0%
4  [  ] 4.9% 8  [  ] 0.7%
Mem[|||||] 4.46G/7.49G Tasks: 120, 447 thr: 1 running
Swp[|||||] 0K/15.9G Load average: 0.74 0.55 0.54
Uptime: 1 day, 02:46:28

PID USER PRI NI VIRT RES SHR S CPUX MEM% TIME+ Command
4836 ben 20 0 4556M 2964M 155M S 31.3 38.7 8h59:18 /usr/lib/firefox/firefox
944 root 20 0 611M 114M 98M S 3.3 11.5 18:59:01 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/r
8538 ben 20 0 4556M 2964M 155M S 3.3 38.7 45:04:07 /usr/lib/firefox/firefox
8539 ben 20 0 4556M 2964M 155M S 2.7 38.7 40:24.90 /usr/lib/firefox/firefox
2416 ben 20 0 1518M 146M 63804 S 2.7 1.9 6:23.52 compiz
6868 ben 20 0 1636M 286M 58748 S 0.7 3.7 9:08.18 /usr/lib/firefox/plugin-container /usr/lib/flashplugin-installer
14667 ben 20 0 34268 5028 3176 R 0.7 0.1 0:00.11 htop
4885 ben 20 0 4556M 2964M 155M S 0.7 38.7 8:31.38 /usr/lib/firefox/firefox
741 root 20 0 4400 1364 1272 S 0.7 0.0 0:12.11 /usr/sbin/acpid
4848 ben 20 0 4556M 2964M 155M S 0.7 38.7 4:53.79 /usr/lib/firefox/firefox
2426 ben 20 0 638M 49840 25376 S 0.7 0.6 0:06.72 /usr/lib/x86_64-linux-gnu/unity/unity-panel-service
13277 ben 20 0 4556M 2964M 155M S 0.7 38.7 0:02.38 /usr/lib/firefox/firefox
12832 ben 20 0 4556M 2964M 155M S 0.7 38.7 0:09.07 /usr/lib/firefox/firefox
4864 ben 20 0 4556M 2964M 155M S 0.7 38.7 3:26.71 /usr/lib/firefox/firefox
12833 ben 20 0 4556M 2964M 155M S 0.7 38.7 0:08.86 /usr/lib/firefox/firefox
8536 ben 20 0 4556M 2964M 155M S 0.7 38.7 3:30.90 /usr/lib/firefox/firefox
2283 ben 20 0 44148 4540 2784 S 0.7 0.1 0:03.36 dbus-daemon -fork --session --address=unix:abstract=/tmp/dbus-b
2372 ben 20 0 512M 31720 21092 S 0.7 0.4 0:04.47 /usr/lib/x86_64-linux-gnu/banfb/banfbdaemon
2616 ben 20 0 1518M 146M 63804 S 0.7 1.9 0:01.70 compiz
2827 ben 20 0 780M 59068 49228 S 0.7 0.8 0:03.30 /usr/bin/nautilus --gapplication-service
4858 ben 20 0 4556M 2964M 155M S 0.7 38.7 0:57.25 /usr/lib/firefox/firefox
4861 ben 20 0 4556M 2964M 155M S 0.7 38.7 0:56.76 /usr/lib/firefox/firefox
3404 ben 20 0 645M 38296 27784 S 0.0 0.5 0:03.36 /usr/lib/gnome-terminal/gnome-terminal-server
5820 ben 21 1 4556M 2964M 155M S 0.0 38.7 5:39.47 /usr/lib/firefox/firefox
5386 ben 20 0 1723M 181M 102M S 0.0 2.4 1:25.35 /usr/lib/libreoffice/program/soffice.bin --writer --splash-pipe=
F1|help F2|Setup F3|Search F4|Filter F5|Free F6|SortBy F7|Nice F8|Nice F9|Kill F10|Quit
    
```

Htop provides a more comprehensive overview of your machine than the standard top utility.

Disable previews
 Many file managers will show a preview of the image files in a directory, but in large folders, this can put a significant extra strain on your computer. If you don't need previews, you can disable them. The process differs a little between file managers, but in *Nautilus*, go to Edit > Preferences > Preview and change the settings.

has to invoke PHP, which then talks to the database – often multiple times – before creating the HTML for the page that is sent to the user. This all happens very quickly, but if a lot of people are requesting pages then this constant processing of PHP and SQL can cause performance problems.

However, a lot of web pages don't change very frequently so there's no need to repeatedly calculate the HTML for a web page. Instead you can perform the processing once and then send the HTML output every time a user requests the page. You only need to update this cached page each time the website changes.

Getting picky

Just as software selection on the desktop can make a huge difference to the performance, choosing the right server software can significantly speed

anything you need it to. However, the downside is that it's memory and CPU intensive. The main culprits in the stack are *Apache* and *PHP*.

The *Apache* web server is powerful, but do you need all the power? If you're trying to get the best performance out of a server – whether this is because you're running a hugely popular website that's struggling to cope with the load or because you're trying to run on low-powered hardware – then an alternative may offer better performance, such as the *Lighttpd* and *NginX* web servers.

Update your software

Web frameworks can be complicated and it's not usually possible to switch between different options easily to save a few MB of memory. There are, however, often tweaks you can perform to increase performance. In many cases, frameworks run on interpreted languages such as *PHP*, *Python* or *Ruby*, and there are changes you can make to the setup of the interpreter that will instantly boost its speed.

Where possible, you should always run the latest stable version of the language. Most programming languages are continually updated with performance (and security) features, so keeping up with the latest version should give you the best performance. The same principal applies to the web framework you're running.

For example, in some cases the *HipHop Virtual Machine* can run *PHP* faster than the standard *PHP* interpreter. Using non-standard software can cause issues with some frameworks, so you'll need to check carefully, but the reward can be a significantly faster server.

Server performance can be hard to understand, and therefore hard to optimise. Only by keeping a close eye on what's going on will you be able to see what's causing problems and what could lead to speedups. The performance monitoring tools we looked at earlier in this article are a good start, but you'll also need specific tools for the environment you run.

The easiest way to increase the performance of a server is to decrease its workload. That might sound like a strange thing to say, but it's often possible to perform the same function while at the same time reducing the

The latest version of PHP should give your web server the best performance

amount the server has to do by judicious use of caching. Take, for example, a web server hosting a *PHP* web app such as *WordPress*. Every time someone visits the site, the web server

up a server as well. The most common Linux web server stack is *LAMP* (*Linux*, *Apache*, *MySQL* and *PHP*) is popular because it's fairly straightforward to set up and is capable of running almost

WEB SPEEDUPS

Computing can be as much about the speed of the web as it is the speed of your PC.

There's little to choose between the performance of any of the major web browsers any more. There is, however, one thing that you can do to speed up your web browsing regardless of which software you choose: use an ad blocker. The exact speedup you get from this varies significantly depending on which sites you look at, but halving page load times is realistic for most browsing. In addition to reducing page load times, blocking adverts can significantly reduce the CPU and memory usage of the browser, which should lead to a better desktop experience. There are a few different ways of blocking adverts:

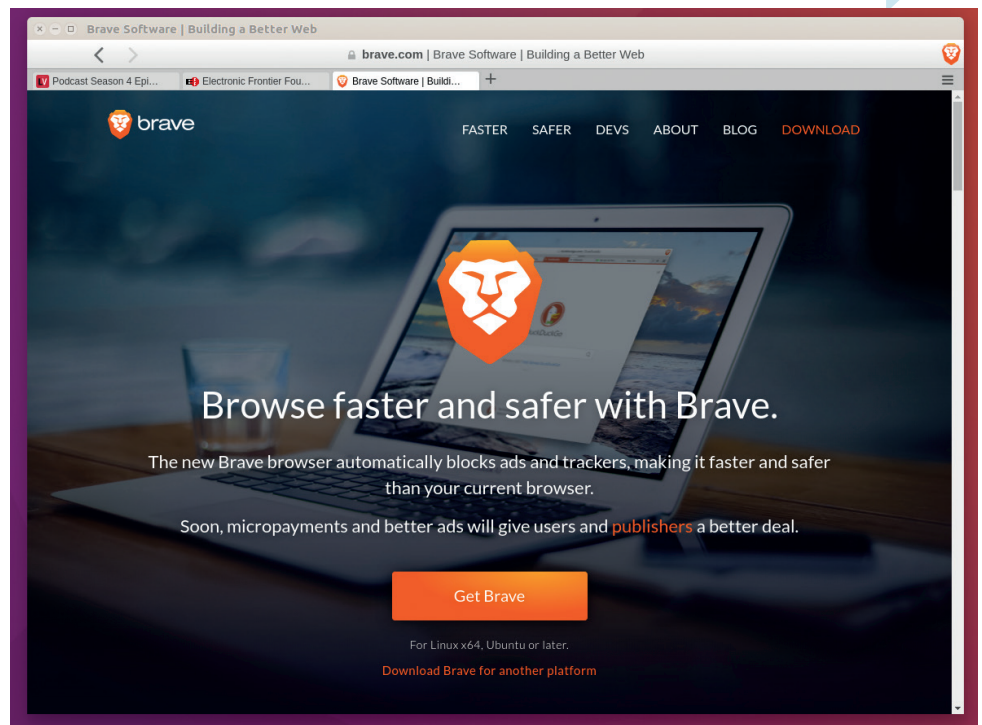
■ Install a browser addon

This is the simplest method for blocking adverts and probably the right option for most people. It's easy to set up and will work on your computer whatever network you're connected to.

■ Use an external ad blocker

You can also use an additional network device to block adverts. This method can run better on machines with limited resources and will automatically block all the devices on a particular network. However, this method will only work on a single network, so it's not appropriate for portable devices such as laptops.

■ Use a browser that blocks adverts



The *Brave* web browser delivers a fast, ad-free experience by default.

The *Brave* web browser has an in-built ad blocker that means you don't need to perform any extra configuration. Future versions of the software will also enable you to pay websites to compensate for their lost ad revenue.

Finally, the plugins you have installed can make a significant difference to browser performance. Flash, in particular, can slow your entire

desktop down. If you don't need it at all, disabling this plugin can speed up your machine, otherwise you can set it to 'click to activate' so that you can control exactly when it's used.

Your Linux box should now be fully tuned for performance, so the only thing left is to decide what to do with all the time you'll save now you don't have to wait around for your PC. 🐉

MOORE'S LAW

Every year, computer manufacturers find a way to get more and more performance out of bits of silicon. There are a few reasons for this, but the primary driver is that chip engineers work out how to make the transistors ever smaller. Smaller transistors mean more of them can fit on a chip, they can be placed closer together and they can run at lower voltages – all things that come together to mean that the chips can be more powerful with smaller sizes. All this is usually summed up by Moore's law (named after Gordon E Moore, co-founder of Intel) which states that the number of transistors on a chip doubles every two years. This is more of a rule-of-thumb than an actual law, but it held true from the mid 70s (when it was proposed) until around 2012. Since then, the speed of

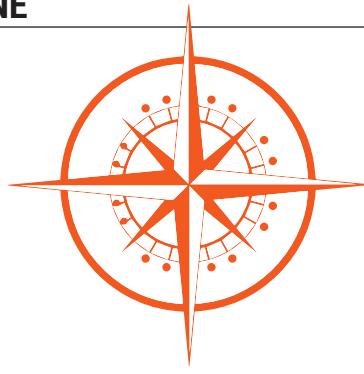
newer machines hasn't held up to the law. There are still other avenues that chip manufacturers can explore to find extra processor speed, so computers are likely to keep getting faster, but at a much slower rate than they have done in the past.

It seems to us that this exponential increase in computer speed should have left concerns about performance back in the 1990s, but it hasn't, and the experience of using a computer day-to-day doesn't seem to have got faster in line with the processing power. We put this question to the man responsible for more low-powered computers than anyone else, the Raspberry Pi foundation's Eben Upton back when the Raspberry Pi 2 came out. This is what he had to say:

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

Eben Upton: It's got a lot to do with Parkinson's law too – work expands to the time available. So this is kind of the CPU version of Parkinson's law. One of the things we've done with the Pi is refuse to accept that 700 million cycles-per-second in the processor is slow. I just won't accept it.

And people kept saying, "It's such a slow processor." and I'm saying "It can do 700 million things a second! Your high-definition screen has only got two million pixels. You can do 350 things to each pixel on your screen every second. How is that slow?"



THE LINUX NEWBIE GUIDE

Are you new to Linux? Or do you want to help your friends and colleagues make the switch? Our guide explains all.

There's nothing wrong with being a Linux newbie. We were all there once. But if you've picked up your first copy of Linux Voice looking for an easy entry into the operating system, or you've been using it for a while but still feel unsure about some things, we're here to help.

To start, what is Linux? It's an operating system, much like Windows and Mac OS X. It runs on your computer, acting as a middle man between your hardware and your applications. It manages your computer's memory, helps different programs to run together (without stepping on each other's toes), and has drivers for your hardware. Historically, Linux had a reputation for being difficult to use, but that's far from the truth now. In fact, you can install Linux and be browsing the web, editing documents and playing games in 15 minutes.

But what makes Linux so awesome? Here are the four biggest selling points:

1 It saves you money

You don't have to pay a penny to use Linux. But how is such a large body of software completely free – who pays for its development? Much of the work on Linux is done by volunteers around the world,

working over the internet. But an increasing amount of work comes from companies such as IBM, Intel, Red Hat and Canonical. They don't make money from selling the operating system, but they generate revenue by offering support contracts, services, documentation and other benefits.

2 It's open to everyone

Because Linux is open source, anyone can study its inner workings. You can download the source code (the original human-readable recipe) of Linux, change it, and recompile it to run on your computer. Now, few people have the technical nous to do this, but it's essential nonetheless: you have

full control over your computer. With Windows, Mac OS or iOS, you can never be sure what the software is doing – you can't get the source code, and you can't fix it yourself, or pay anyone else to do it.

3 It's super reliable

Configured correctly, a Linux system simply won't crash unless something is wrong with your hardware. We know people who've been running Linux servers for several years without a single reboot. Linux is designed in such a way that its various components are well isolated from one another, so if there's an issue with one part of the operating system (such as the graphical user interface), the rest of it carries on chugging away – for years and years.

4 It works with your files

Although Linux is a different operating system to Windows and Mac OS, and doesn't run all of the same programs, it's the most compatible OS in existence. You can open your *Microsoft Office* documents in *LibreOffice*, you can play all your videos and music in *VLC*, and there are Linux equivalents for pretty much every application on Windows and Mac OS X.

Linux experts:
this guide is Creative
Commons (BY-SA)
licensed, so cut it out
or photocopy it and
share it. Help others to
convert!

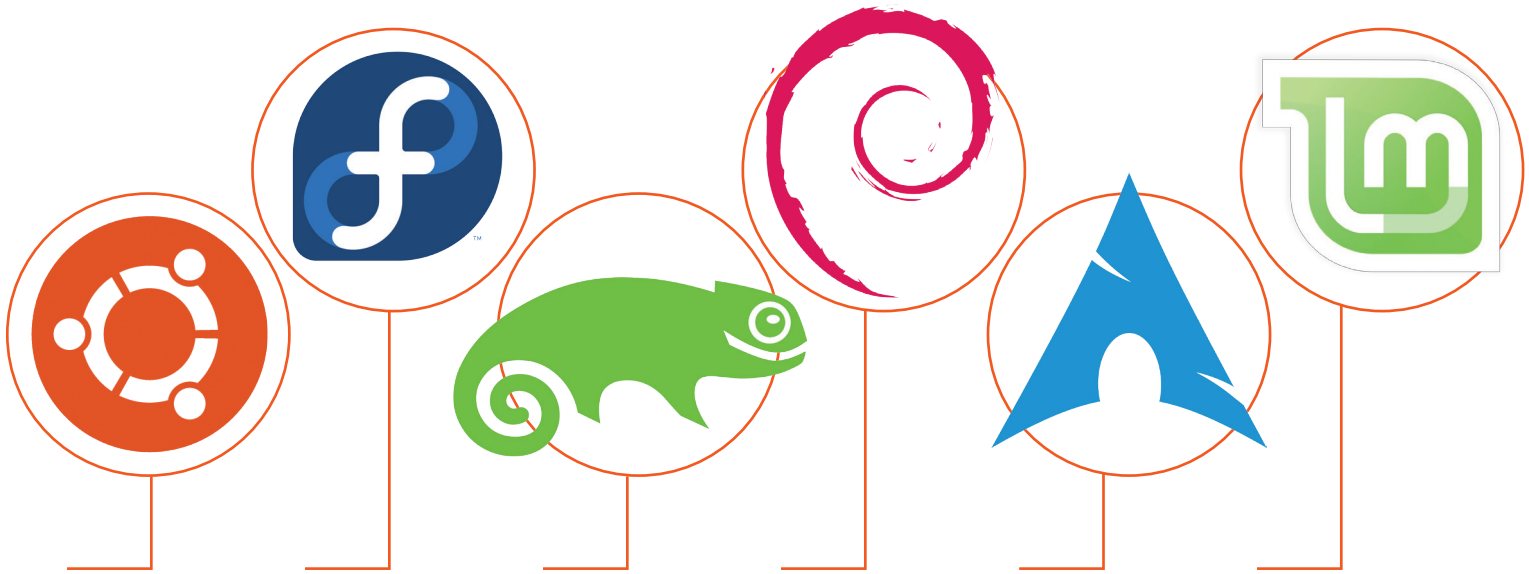
CHOOSING A DISTRO

Linux comes in many flavours – find the best one for you.

Before you get started on your Linux journey, you need to choose a distribution – a bundle of Linux and related software. This might seem like an extra hassle when compared to the Windows or Mac OS approach, but it makes sense. Some Linux distributions are

targeted at newbies, some at power users, some at security, some at low-end devices (such as old netbooks), and so forth. They all share the same core software, but include other features and add-on software to be the most suitable for a specific scenario.

Now, there are hundreds of Linux distributions out there, which can seem baffling at first. But the vast majority are simply based on another one, so in actuality, there's only a handful of unique distributions. Let's look at a few of the most notable...



Ubuntu

www.ubuntu.com

Ubuntu Linux is primarily geared towards desktops and laptops, although it's making gains on tablets and phones as well. With Ubuntu, you can get a modern, well-tested version of Linux with just a few mouse clicks.

Fedora

www.getfedora.org

Fedora is a community supported distribution known for incorporating cutting-edge technologies, and makes new releases every six months. Like Ubuntu, Fedora focuses on having an attractive and versatile interface.

OpenSUSE

www.opensuse.org

OpenSUSE started life in the mid 90s. It's popular with intermediate Linux users, sporting an excellent configuration tool called Yast that lets you tweak all aspects of your system from a single point.

Debian

www.debian.org

Debian is well known for its stability and is therefore used on tens of millions of servers around the world. Debian is a community project and provides the basis for many other distributions, such as Ubuntu and its spin-offs.

Arch Linux

www.archlinux.org

Instead of having big updates every six months or every year like most other distros, Arch constantly changing with the latest software. This is great for power users, although it can cause some problems.

Linux Mint

www.linuxmint.com

Linux Mint is based on Ubuntu, but provides a different interface and set of default software. It's popular amongst new users and has a very helpful supporting community.

Making the big decision

So after all this, we recommend going with Ubuntu. It's the best known distro, is very polished, and has a huge supporting

community on the web (eg www.askubuntu.com). The Ubuntu team puts a lot of effort into its interface and into making sure that the operating system works well out of the

box, so we think it's the best way to start. After a few months of working with Ubuntu, you'll be confident enough in Linux to try other distributions.

GIVING CREDIT TO GNU

Here's an important history note: what we call "Linux" today is the work of multiple projects that have been running since the 1980s, all of which have worked together to create a free, open and shareable computing platform. One of these projects dates

back to the early 1980s, and is called GNU, for GNU's Not Unix (recursive acronym glory).

GNU developed a lot of software to help create a fully free operating system, and was paired up with the Linux kernel in 1991 to create something

everyone could install and use. So the GNU project played a huge role, which is why you sometimes see Linux referred to as GNU/Linux, and today the operating system has hundreds of thousands of developers around the globe.

SYSTEM REQUIREMENTS

- 1GHz Intel/AMD CPU
- 2GB RAM
- 10GB drive space

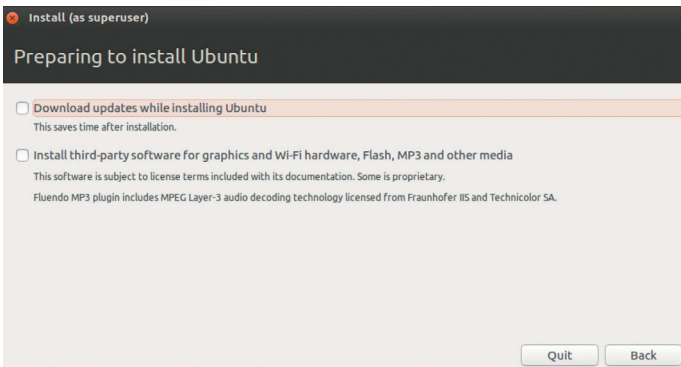
INSTALLING UBUNTU LINUX

Follow our step-by-step guide and get using Linux in 15 minutes...



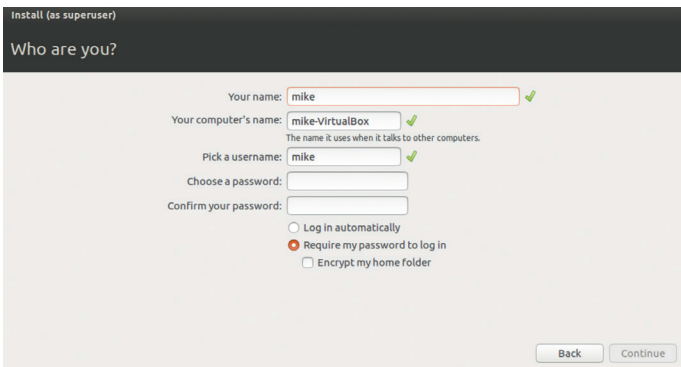
1 Get Ubuntu

Point your browser at www.ubuntu.com/download/desktop and get the latest version (eg 16.04). You will download an ISO file, which is a disc image that can be burned to a DVD-R using your regular disc burning software (you can also create a bootable USB stick).



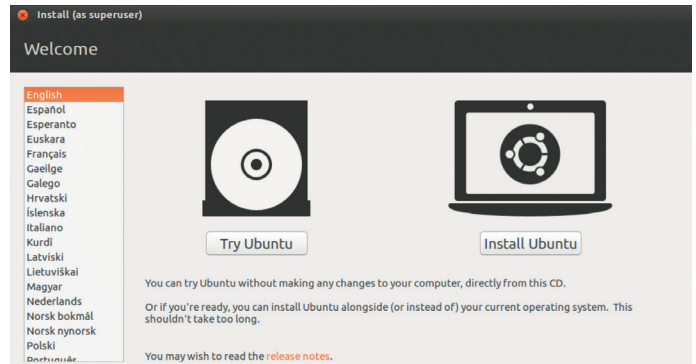
3 The installer

Ubuntu's installer will check that your PC has sufficient space to install Linux. If you are connected to the internet, you can download updates, extra drivers and media file codecs during installation; click the icon in the top-right to set up Wi-Fi connection if you need one.



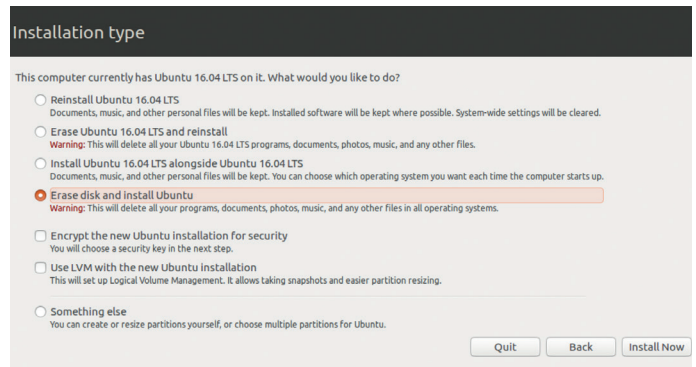
5 Create a user account

The Linux files will be copied to your hard drive, and you'll be asked to set your location and keyboard layout. You will also be prompted to set up a user account so that you can identify yourself to the operating system and log in – don't forget your password!



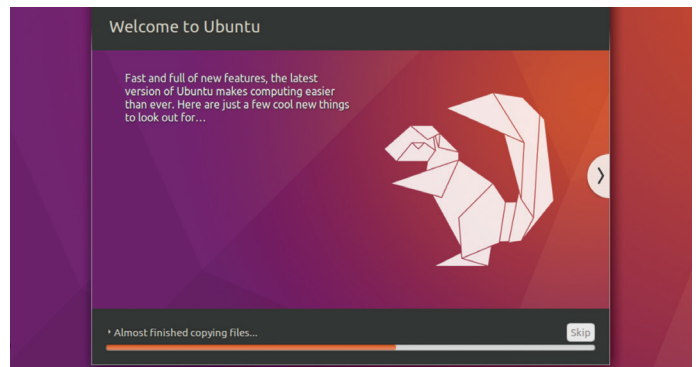
2 Start your PC

Next, you need to boot up your PC from the DVD-R or USB key; you normally need to press a key on your keyboard when your computer starts to do this, so consult your PC's documentation to find out how. After a few moments, the screen above will appear.



4 Splitting the disk

In the next step, choose where to install Linux on your hard drive. You can install it alongside Windows, and have a menu when you start your PC to choose your operating system, or you can dedicate the whole hard drive to it.



6 Almost finished...

Grab a cup of tea, and when all the files have been copied over, the installer will prompt you to reboot the machine. Click on Restart Now and remove the DVD or USB key once the PC restarts, then choose Ubuntu from the boot menu that appears.

USING LINUX

Now you're ready to work (and play!) in your new Linux installation.

When Ubuntu first starts, a window will appear listing some common keyboard shortcuts that are worth memorising to make you work more quickly, so once you've glanced through them, click the X button in the top-left. Next, click on the Ubuntu button in the top of the bar on the left. This is similar to the Start button in Windows: it lets you browse included software (go to the Applications button at the bottom after clicking it, and then Installed to see what's included by default). You can also type to search for files or to run programs directly.

Other buttons in the panel on the left are shortcuts to useful programs. When you start a new program, its icon will appear on this bar; right-click it and choose "Lock to Launcher" to keep it there after closing the app. In this way, it's a bit like the Windows taskbar – but much more flexible.

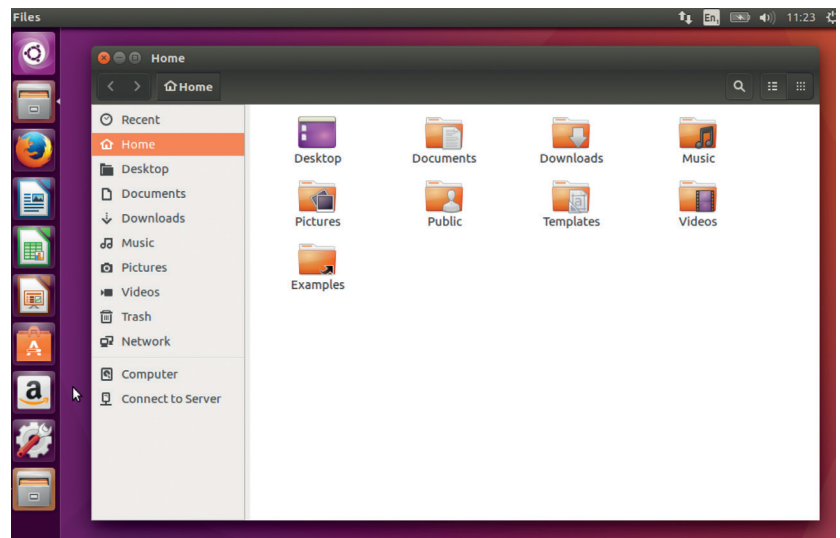
When you're running applications, you can click and drag the titlebars to move them, and use the edges to resize them. Ubuntu has a global menu bar, like in Mac OS X; when using an application, move your mouse pointer to the top bar to show menu entries. Also in the bar at the top you'll find icons for audio levels, power management and networking. Click the cog icon on the far-right to log out or shut down the machine when you're finished with your work.

To change settings with your installation, click the cog-and-spanner icon in the panel on the left. This opens up the Systems Settings window, from which you can configure your installation, manage hardware, and add new user accounts. If multiple people will be using Linux, give them all separate accounts so they can have their own desktop and software settings.

Included software

To access your personal files, click on the drawer button underneath the Ubuntu button on the left-hand panel. Your "home" directory is like My Documents in Windows – it's where your personal files are stored. If you insert a DVD or plug in a USB key, a window will pop up showing its contents, and on the left-hand panel of the file manager, you can also access resources on the network.

Underneath the drawer button you'll see an icon for *Firefox*, a web browser you're probably familiar



with from Windows or Mac OS. *Firefox* is arguably the best browser out there, combining good performance and thousands of extensions with excellent privacy settings. And underneath *Firefox* you'll see three icons for *LibreOffice*, opening the word processor, spreadsheet and presentation tool respectively.

LibreOffice is the flagship office suite on Linux, and is tremendously capable, having seen decades of development in its previous incarnations as *OpenOffice* and *StarOffice*. *LibreOffice* does a very good job of opening *Microsoft Office* documents – although there can be slight formatting issues with some very complicated documents. Still, if you open an *Office* document from one version of the suite in a different version, you'll likely experience the same thing, so this is something even Microsoft struggles with!

For email, click the Ubuntu button and search for *Thunderbird*. This is an email client from the makers of *Firefox*, and is mature and very stable. Other pre-installed software worth exploring is *Rhythmbox* (a music player), *Empathy* (for instant messaging) and *Shotwell* (a photo manager). Of course, you'll find plenty of small tools such as a calculator and text editor as well.

So, those are the basics – have fun taking it from here! If you need any help, visit our forums at

www.linuxvoice.com 

Ubuntu's Unity desktop is friendly for newbies and liked by many power-users as well.

ADDING MORE SOFTWARE

Your Ubuntu installation comes pre-installed with many top-class applications, and there are thousands more available. Click on the Ubuntu button, type "software" and choose the Ubuntu

Software Centre to explore programs available to download – most of them free and open source. You can browse categories down the left. Some of our recommendations include *Gimp* (an image editor),

Audacity (for editing audio files), *OpenShot* (a movie maker), *VLC* (a media player that handles virtually every format under the sun) and *HomeBank* (a personal finance tool).

SECRETS OF CALIBRE

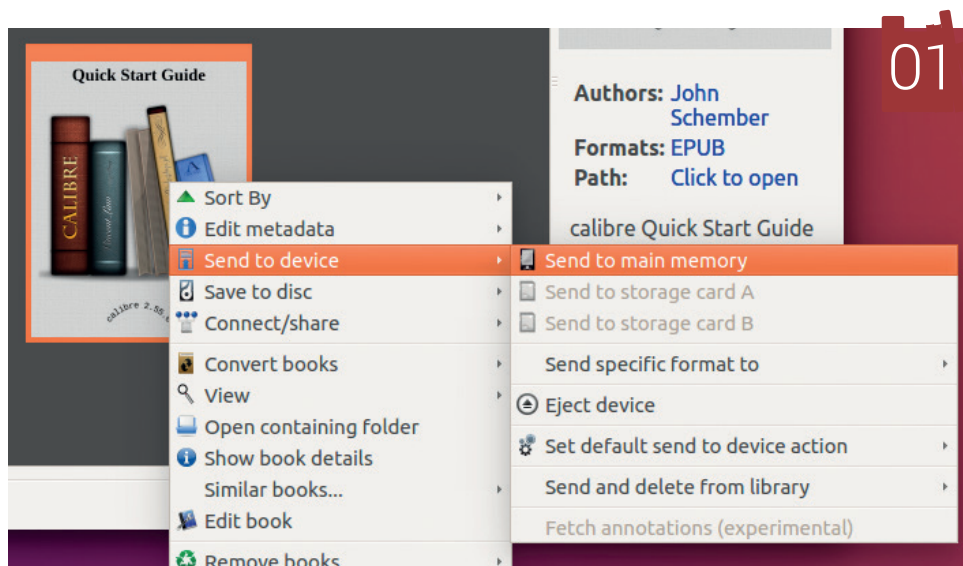


Master your eBooks and create a flexible personal library.

The technology used to make them has changed hugely from hand-written papyrus to printed paper to computers and the very latest eInk screens. Books have not only survived all this change but they've grown ever more popular with each technological advance. The latest incarnation of this ancient format – eBooks – provides some new challenges to the reader, such as how to store

and view books when they're reduced to data files, but eBooks also allow instant access to a mind boggling number of books.

When it comes to eBooks on Linux, there's one tool that stands out above the others: *Calibre*. It's our application of choice for reading, managing and even creating eBooks. It's full of features, but can be confusing to new users. Here are its eight best features...



01 Syncing While you can read eBooks on your PC, you'll usually get a better experience on an eReader, a phone or a tablet. After all, it's nicer to relax in a comfortable chair with an eReader than to sit at your desk. Calibre enables you to send books to devices that are connected (either physically or in some cases via a local network). This way you can keep your full library on your PC (where you probably have more

storage and a better backup system), and just keep the books you're currently reading on your devices.

02 Convert There are a number of different eBook formats, and not all eReaders can support all formats. Fear not: *Calibre* can step in to convert your library from almost any format to almost any format. However, some eBook stores include Digital

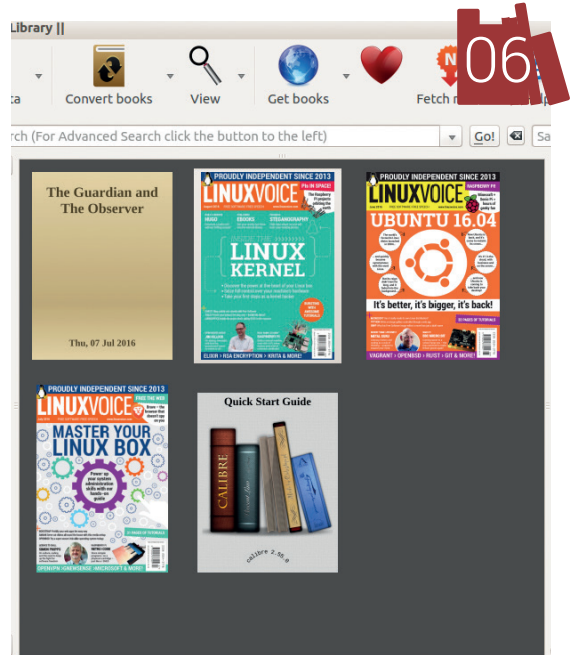
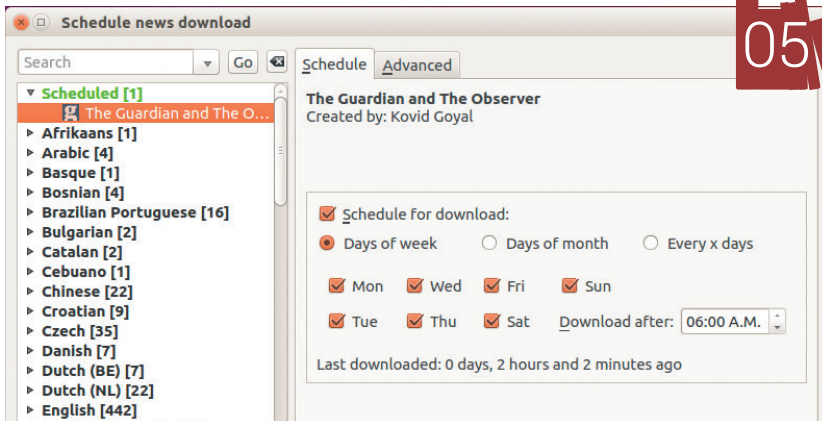
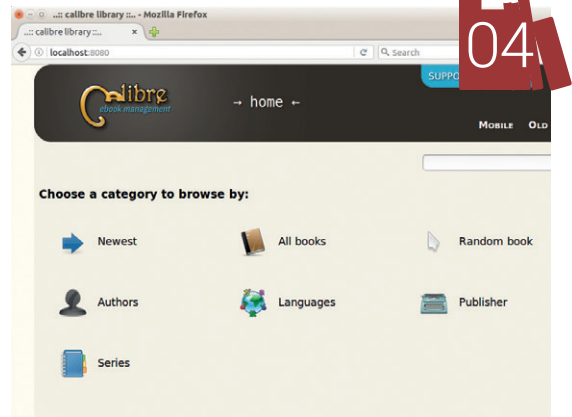
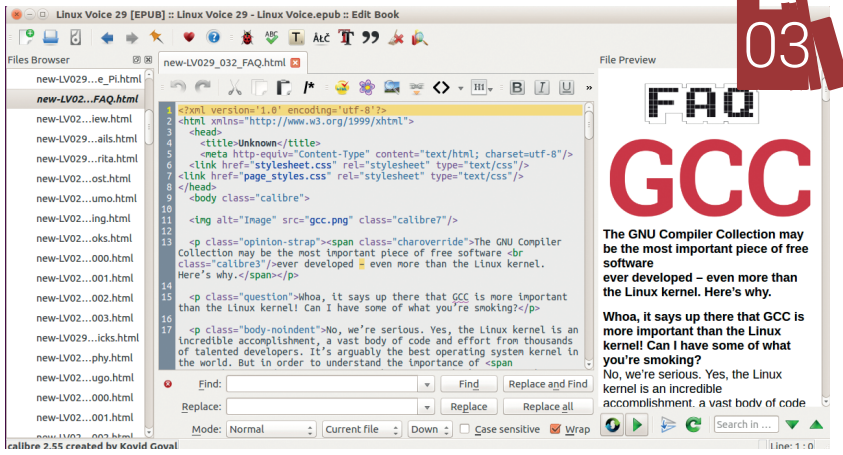


Rights Management (DRM) on their downloads, which limits what you can do with your eBooks. *Calibre* will not convert these books unless you use a DRM-remover tool first (which may be illegal in some countries).

03 Editing As well as reading eBooks, *Calibre* helps you create your own. The powerful in-built editor enables you to tweak existing books or write new ones from scratch. You'll need to understand HTML to make the most of it, but for most eBooks, a simple layout is best so you only need basic HTML. It's our tool of choice for creating the Linux Voice digital editions.

04 Web server *Calibre* includes a built-in web server. Go to Preferences > Change Calibre Behaviour > Sharing Over The Net to set it up. Once it's running, you can access your books through the web interface. This makes it easy to transfer books onto devices without the need to

If your Calibre machine is available on the public internet, you can get access to your library from anywhere in the world




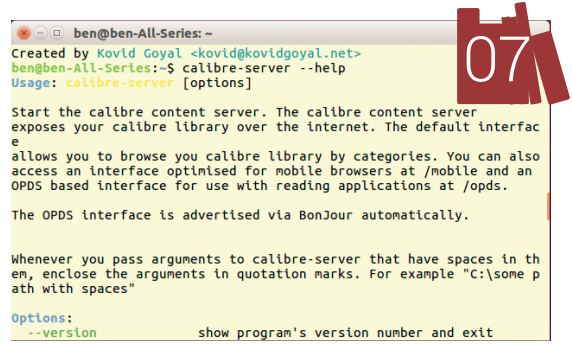
physically connect them to the machine. If your *Calibre* machine is available on the public internet, you can get access to your library from anywhere in the world.

05 Online news
Calibre will automatically download online news sources and convert them into eBook formats that you can browse offline. You can schedule these downloads to happen at a certain time, for example, first thing in the morning, so you have something to read on the train to work. If coupled with the web server (secret 4), this could give you your own private stash of downloadable reading matter.

06 Library management
 If you're an avid reader, you could quickly find yourself swamped with more books than you can easily manage. *Calibre* helps you filter your library by author, series, rating, format, tags and other options. Used properly, this filtering should mean that you never have to spend long looking for what you want regardless of how many books you have.

07 Command line interface
 While *Calibre* is primarily a graphical application, you can use it from the terminal. This could be particularly useful if you wanted to run it on a headless server to enable you to access your eBooks and scheduled downloads when your main PC is switched off. You can also use this interface to convert eBooks between formats or to launch the viewer – both features that could be useful if integrating *Calibre* in scripts or other bits of software.

08 Tweaks and plugins
 The out-of-the-box *Calibre* setup is good for most people, but you can add features to make it fit your personal workflow. Plugins enable you to add new bits of functionality, such as the ability to access online stores or convert to an obscure file format. Tweaks enable you to change minor bits of *Calibre* behaviour by changing some of the underlying Python code. Between these two sets of tools, you can make *Calibre* work exactly how you want it to. 



BUILT WITH

TM



ARDUINO

Les Pounder looks into the little machine that's powering a quiet revolution – the march of the makers!

The Arduino microcontroller has changed the world. It's now part of the larger Maker ecosystem, which it helped found and later shape. The Internet of Things, a network of connected appliances and devices, Physical Computing, where computer science concepts are used with hardware in new and interesting ways: without the Arduino we would not have these technologies and communities and technologies as we know it.

But how did the Arduino come to life? Well, it all started as a way for artists to integrate technology

into their work bringing forth the merger of arts and technology. In the mid 2000s the maker community as we know it was still in its infancy. The Interaction Design Institute Ivrea (IDII) in Italy, a centre specialising in how users interact with computers, was working on creating a development platform with a supporting hardware device, called Wiring, which formed part of Hernando Barragán's master's thesis in 2003. The first prototype Wiring project used the Parallax Javelin Stamp microcontroller, but this



The Arduino started as a way for artists to integrate technology into their work, bringing forth the merger of arts and technology

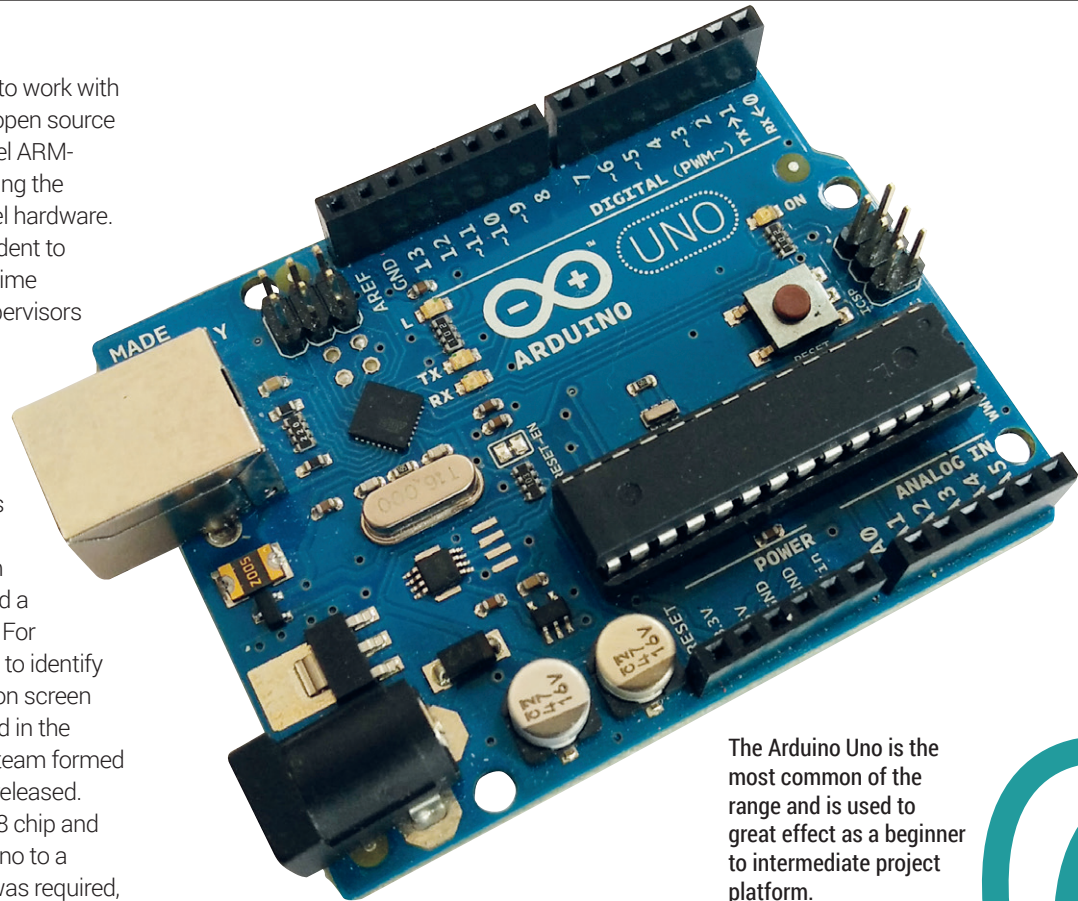


microcontroller required proprietary tools to work with the board; Hernando wanted to make an open source project, and so later prototypes used Atmel ARM-based AT91R4008 microcontrollers, starting the popular combination of Arduino and Atmel hardware. Hernando later went on to be the only student to graduate with distinction in 2004. At this time Massimo Banzi and Casey Reas were supervisors for Hernando's project and took part in a four-week project called "Strangely Familiar" that taught physical computing to 22 students.

The goal of Wiring, and the earlier Processing project created by Casey Reas and Benjamin Fry, was to enable non-programmers, typically artists, to program using an easy-to-understand language and a simple open source prototyping platform. For Processing, the term "sketches" was used to identify the code of a project which would create on screen visualisations. This term has been retained in the Arduino community. In 2005 the Arduino team formed and the first Arduino-branded board was released. The Serial Arduino came with an ATmega8 chip and ran at 16MHz. To connect the Serial Arduino to a computer a DE-9 9 pin serial connection was required, a port that is no longer found on many computers. The Arduino USB was the first board to feature a USB interface and this was handled via the FTDI FT232BM, which enabled serial data and power connections to the host computer.

Number 1

The most famous Arduino is the one that everyone starts with: the Uno. The Arduino Uno is a humble platform that serves as your first step to becoming a maker. Coming with an Atmel ATmega328P microcontroller running at 16MHz and 32kB of flash memory, the Arduino Uno has 14 digital pins, with six of these pins also coming with pulse width modulation (PWM), which useful for controlling motors with a variable speed. The Arduino Uno, as well as other Arduinos, comes with six analog pins,



The Arduino Uno is the most common of the range and is used to great effect as a beginner to intermediate project platform.

something that the Raspberry Pi does not feature. These analog pins can be used with potentiometers to create delicate forms of input for precise control of an output device, such as a motor or LED.

Over time the Arduino project went on to become the *de facto* standard for makers and hackers, and many books, projects and accessories were produced. The Arduino's popularity has also seen the creation of "Arduino Day", which for 2016 took place on 2 April. The Arduino has been used by makers and artists for installation pieces such as Minimaforms' Petting Zoo, which looked at how environments can interact with users to shape the architecture of the future. The low price and the ease of use is what incites makers and hackers to choose the Arduino for their projects.



SEND IN THE CLONES!

Being an open source platform, the Arduino has spawned many clone devices. Due to rules put in place early in the life of the Arduino, no clones may be called an "Arduino"; rather they are "Arduino Compatible"

These Arduino-compatible boards are just as good as their official counterparts, and in most cases are significantly cheaper. We managed to source an Arduino Uno clone for £1.68 including postage from AliExpress, and we found an Arduino Mega (the larger board with more pins) for a mere £4.08. So should we buy these boards in favour of the official boards? From our personal opinion purchasing an official board as your first device will provide you with the confidence that it has been built and tested to a high standard, ensuring that you have a good start in the

world of physical computing. Once you become confident with the platform and start creating multiple projects, it would be financially prudent to purchase the clones; just make sure that they are of good quality. Open source hardware is generally good quality, but sometimes a rogue board slips through the net.

One Arduino compatible that is worth attention is The Shrimp project, based in Morecambe and named for the famous shrimping farming community. The Shrimp is a build-it-yourself bare-bones Arduino that comes as a kit and is assembled on a breadboard. They work exactly like an Arduino Uno, and retail for £10. These kits help makers understand the parts that make up these powerful prototyping boards. More information from <http://start.shrimping.it>.



Support Center Weather in your city Sign In Sign Up °C °F

OpenWeatherMap Weather Maps API Price Partners Stations News About

Weather in your city

Blackpool, GB 14.5 °C

Rain get at 2016.05.11 18:01 (Wrong data?)

Wind	Moderate breeze 7.46 m/s East-northeast (68.5004)
Cloudiness	overcast clouds
Pressure	1011.83 hpa
Humidity	81 %
Rain	0.545
Sunrise	5:17
Sunset	21:2

Main Daily Hourly Chart Map Satellite

Next hours

11 May 22:00 Precipitation: 0.1 Temperature: 12.8

19:00 22:00 01:00 04:00 07:00 10:00 13:00 16:00 19:00 22:00 01:00

14.28C 12.81C 12.01C 11.14C 11.86C 15.38C 17.98C 18.5C 17.55C 14.66C 12.04C
7.71m/s 7.46m/s 6.31m/s 5.66m/s 5.46m/s 6.46m/s 6.91m/s 7.36m/s 7.01m/s 6.46m/s 6.26m/s
1013.78 1014.26 1014.82 1014.46 1014.36 1014.84 1014.82 1014.57 1014.82 1016.72 1018.11

The open source hardware of the Arduino can be broken down to its constituent elements and used on a breadboard, such as the ShrimpingIt project.

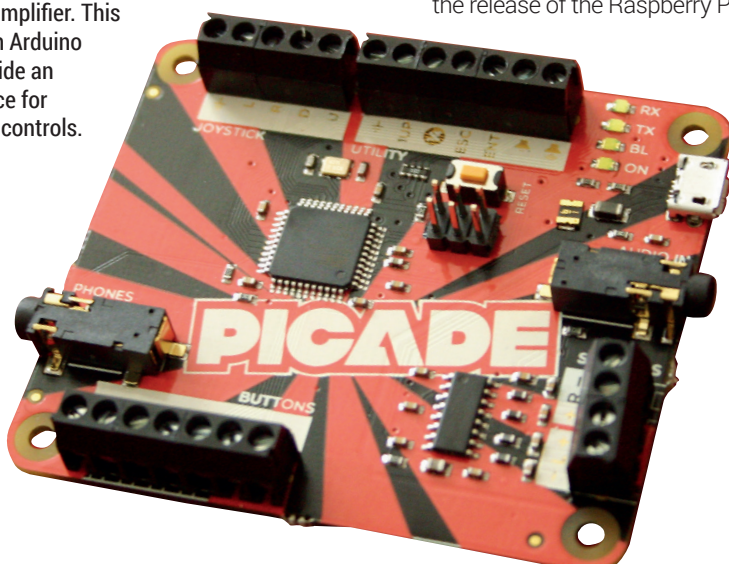
In recent years we have seen an explosion of single-board computers taking on the dominance of the microcontroller-based Arduino. The Arduino has

which is now in its fourth year of production having seen a number of iterative releases that have added more to the board for the same price of \$35. Not

“ In recent years we have seen an explosion of single-board computers taking on the dominance of the microcontroller-based Arduino



Arduinos turn up in the most unlikely of places. Here we see a Picade joystick controller and audio amplifier. This uses an Arduino to provide an interface for arcade controls.



been with us since the mid 2000s and has driven the uptake of physical computing and powered the maker movement, and this is long before the Raspberry Pi, Microbit etc. The Arduino's position as the leading platform for hardware hacking was threatened by the release of the Raspberry Pi,

resting on their laurels, the Arduino team have also released new boards. In 2013 we saw the release of the Arduino Yun, powered by an ATmega32U4 and including an Atheros AR9331 Wi-Fi System on a Chip (SoC) enabling the Yun to use Wi-Fi in projects. The Yun also provides a 400MHz processor and 64MB of RAM to use with Linino, a derivative of the OpenWrt Linux distribution. The Yun was not alone in heralding new features to the Arduino platform: in late 2013 the Intel Galileo was released, which was the first Arduino-compatible board to feature an Intel processor. The Galileo was designed to be hardware- and software-compatible with Arduino Uno projects while offering greater processing power, thanks largely to a 400MHz Intel Quark SoC X1000, a 32-bit single-core processor that offered the same processing power as an Intel Pentium from the mid 2000s.

The Intel Galileo also offered a greater number of ports and connectors for the growing Internet of Things movement, namely Ethernet, PCI Express, micro SD cards and USB 2.0. The board also offered the ACPI power-saving functionality to efficiently

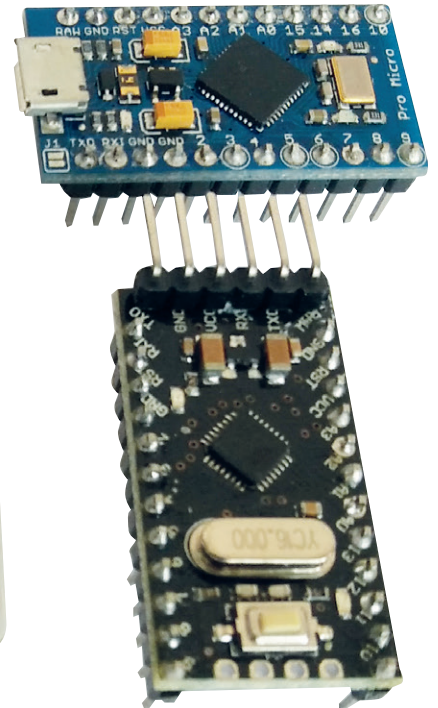
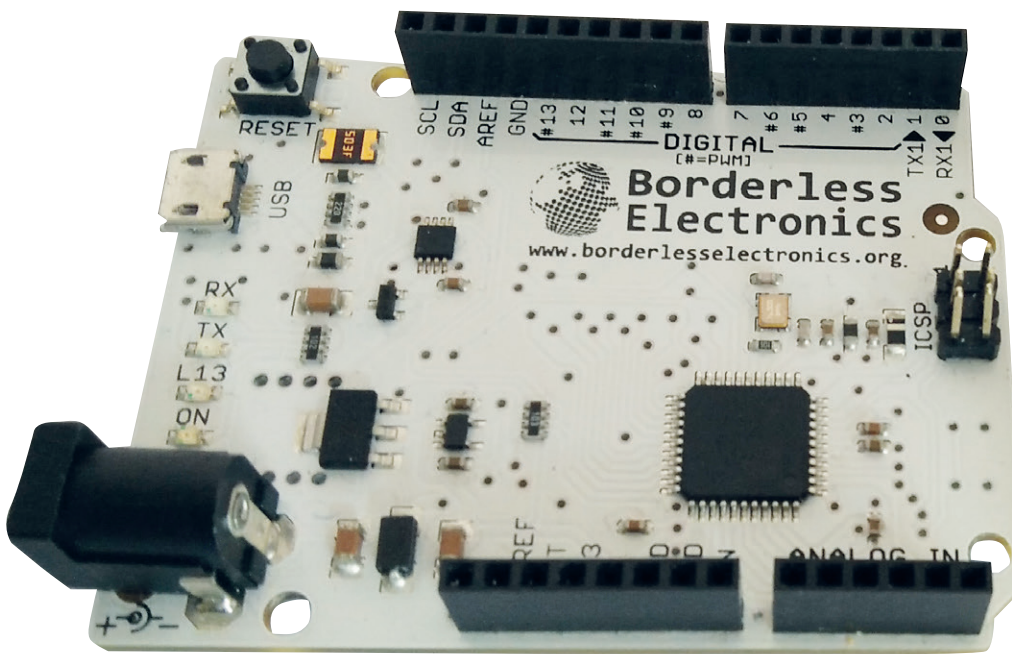


manage the power of your project. The Galileo later spawned its successor, the Galileo Gen 2, which offered the same functionality but in a slightly more refined platform.

The Internet of Things is something that we have mentioned throughout this feature, and the Arduino product line has seen its latest board, the MKR1000, offer a new and unique platform. Powered by an ARM Cortex M0 and featuring a low-power Wi-Fi chip, the MKR1000 is seen as an IoT platform to power the next generation of smart devices. The board can still be programmed using the familiar Arduino editor.

As of May 2015 the Arduino is also known in some sales territories as Genuino, a trademark created by four of the original five founding members who initially formed Arduino LLC, a company that would hold the trademarks for the brand and license the manufacture and sale of boards to external companies. This occurred as a result of the Arduino trademark being secretly registered in Italy by a fifth member of the team. Subsequent negotiations failed to unite the brand, forcing the Arduino LLC team to create the Genuino brand for use outside of the United States of America.

There are many clone Arduino boards on the market. Some match the Arduino specification, whereas others can be directly inserted into a breadboard.



A SIMPLE TUTORIAL TO CREATE A SENSOR THAT SHOWS DISTANCE USING LEDs

There is no better way to understand how easy the Arduino is to use than by getting hands-on with it. In this project we introduce the Arduino platform by creating a distance sensor that uses an ultrasonic sensor to measure distances using a pulse of sound. This is then processed and output via a series of LEDs.

Hardware setup

We start the project by constructing the circuit. We used an HC-SR04 ultrasonic sensor that we purchased from eBay. This requires four connections: to 5V power; ground (GND); the trigger, which sends a pulse; and an echo, which receives the reflected pulse. As you will see in the diagram we have opted to use the same pin (pin 12) on the Arduino. Using code we can switch the pin from an output, which will send the

pulse, to an input which will receive the echo. Please refer to the diagram for more information.

Software setup

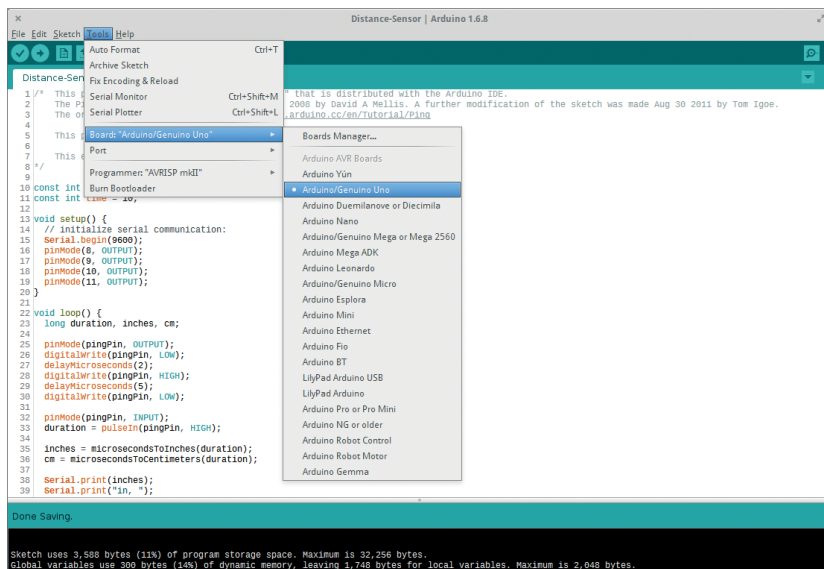
Our first step is installing Arduino on our machine. The Arduino homepage has complete instructions on how to do this for all operating systems: <https://www.arduino.cc/en/Guide/HomePage>.

With the Arduino software installed, launch the application and you will immediately see a blank screen with today's date as a suggested filename. In here we shall write our code.

We start by creating two variables that are constant integers; in other words these values do not change. These variables are **pingPin**, used to state which pin our Trigger and Echo pins are connected to, and **time**, which is used to store a default time value to control

You will need

- An Arduino Uno
- Four LEDs
- Four 220Ω resistors (red-red-brown-gold)
- A breadboard
- Male-male jumper wires
- All of the code for this project as well as a high-resolution diagram can be downloaded from <https://github.com/lesp/Arduino-Distance-Sensor/archive/master.zip>



The Arduino IDE works with all versions of the Arduino boards. If a board is not listed, it can be added via the Boards Manager option. This includes the ESP8266-based boards.

the pace of the code.

```
const int pingPin = 12;
const int time = 10;
```

In our next block of code we set up the pins that will be used for our LEDs. These pins are 8–11 and each one is an output. We also start a serial interface at 9600 baud (bits per second). We will use the serial monitor to check that our code works.

```
void setup() {
Serial.begin(9600);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
}
```

We now move to the main loop of our code. This loop will continue forever. We start the loop by declaring long variables, used to contain long numbers, for the duration, and lengths in inches and cm. Next we turn our **pingPin**, pin 12, into an output before we ensure that the pin is turned off. We then pause for 2 microseconds before turning the pin on for 5 microseconds, which is just enough time to send a pulse. We then turn the **pingPin** off.

```
void loop() {
long duration, inches, cm;
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
```

Still inside the loop we now change our **pingPin** from an output to an input, ready to receive the echo ping. We then store the duration of time taken in a variable that is then used to calculate the distance in inches and centimetres. The conversion process is handled later in the code.

```
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
inches = microsecondsToInches(duration);
```

```
cm = microsecondsToCentimeters(duration);
```

Our next section of code is a method of debug. We print the distances to the Serial Monitor, accessed via the Tools menu, so that we can see the values and check that they are correct.

```
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
```

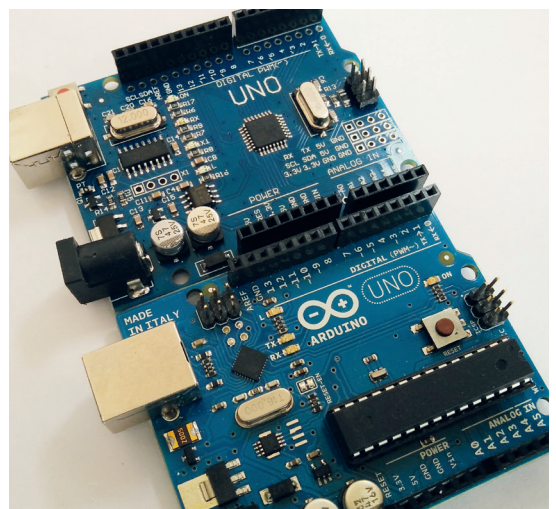
We now start a conditional test that will check the distance returned, and use that to control our LEDs. We will show a portion of this code, the whole of which can be downloaded from our GitHub page.

Our first test is to see if an object is less than 5cm away; if that is True, we print "Less than 5CM" to the serial monitor. We turn pin 11 on thus lighting our LED. Pins 10,9 and 8 remain turned off. We then delay for 10 microseconds using the **time** variable we created earlier.

```
if (cm < 5){
Serial.print("Less than 5CM");
digitalWrite(11, HIGH);
digitalWrite(10, LOW);
digitalWrite(9, LOW);
digitalWrite(8, LOW);
delay(time);
}
```

Our second test handles distances less than 10cm and greater than 5cm. If that is the case then all pins except pin 10 are turned off, thus illuminating a new LED. We use **else if** for this condition. If you are used to Python, this is referred to as **elif**.

```
else if (cm < 10 && cm > 5){
Serial.print("Less than 10CM");
digitalWrite(11, LOW);
digitalWrite(10, HIGH);
digitalWrite(9, LOW);
digitalWrite(8, LOW);
delay(time);
```



There are few differences between clones and the original Arduino (chiefly they cannot use the Arduino trademark), but the clones can share the same layout as the originals.


```

Distance-Sensor | Arduino 1.6.8
File Edit Sketch Tools Help
Distance-Sensor
1 /* This project uses the built in example "Ping" that is distributed with the Arduino IDE.
2    The Ping example sketch was created on Nov 3 2008 by David A Mellis. A further modification of the s
3    The original code can be found at http://www.arduino.cc/en/Tutorial/Ping
4
5    This project featured in Linux Voice magazine
6
7    This example code is in the public domain.
8 */
9
10 const int pingPin = 12;
11 const int time = 10;
12
13 void setup() {
14   // initialize serial communication:
15   Serial.begin(9600);
16   pinMode(8, OUTPUT);
17   pinMode(9, OUTPUT);
18   pinMode(10, OUTPUT);
19   pinMode(11, OUTPUT);
20 }

```

There are three more conditions to test, which you can see in the code download for this tutorial. Our final condition handles when the distance is greater than 30cm. All of the LEDs are turned off and we print a message to the serial monitor.

```

else {
Serial.print("Greater than 30CM");
digitalWrite(11, LOW);

```

```

digitalWrite(10, LOW);
digitalWrite(9, LOW);
digitalWrite(8, LOW);
delay(time);
}

```

We now close the loop after delaying the code for 100 microseconds.

```

delay(100);
}

```

Our final lines of code are long variables that we use to store the distances when calculated as inches or centimetres.

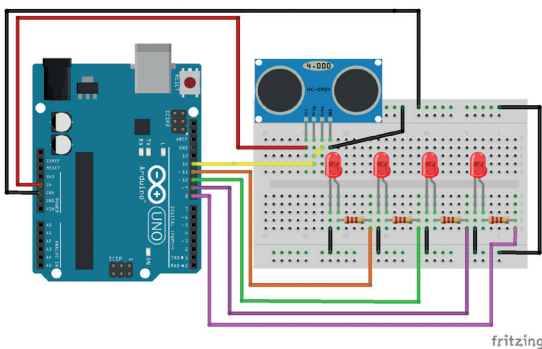
```

long microsecondsToInches(long microseconds) {
return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
return microseconds / 29 / 2;
}

```

With the code completed, save your work. Ensure that your Arduino is connected to your computer and then click on the Upload button (an arrow pointing right). Once it's done uploading, the Arduino will reboot and the script will run. Place an object in the path of the sensor to trigger the LEDs. 📺



Building the circuit for our project is relatively simple; just take it step by step and follow each wire from point to point.



SHIELDS UP CAPTAIN?

The Arduino platform has an extensive ecosystem of peripherals and components that can be added to your project, from simple components such as LEDs and buzzers to more bespoke add-ons such as GPS and 3G data. The Arduino uses a common add-on system called shields. These shields are placed on top of the Arduino and provide extra functionality.

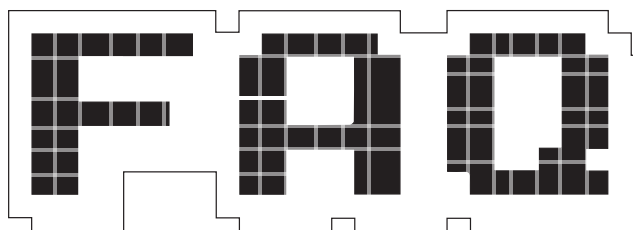
For example, the Ethernet shield provides a simple Ethernet interface, enabling our Arduino to work as a low-powered web server, albeit not one that you would want to power your site. The main use for the Ethernet Shield is to send data to the web, so sensor data can be gathered and sent to a remote computer for processing. Other notable shields are the Adafruit Wave Shield

and SparkFun MP3 Shield, which provide audio output for your projects – imagine a sensor-triggered scare device for this year's Halloween party. Arduino shields come from a variety of sources and retail from around £5 to £50 and as with other elements of the Arduino community, many of these shields can be picked up cheaper as clones but with varying levels of quality.

When starting out with the Arduino you may be tempted to splash out on shields, but before you do, get used to the Arduino as it is. You will be amazed as to how much can be done with such frugal resources. We don't need a GHz CPU and gigabytes of RAM to run a sensor-powered data collection device; rather we need a dependable and power efficient platform to work from.

The Arduino IDE is the default way to work with your board. It comes with a series of examples and tutorials designed to help new users get to grips with the Arduino.





Flatpak

Hurrah! Distro-independent packaging is finally here, after so many years of waiting. Or is it?

MIKE SAUNDERS

Q Oh great, yet another packaging format! Is that what the world really needs?

A We know what you're thinking. We love the flexibility and choice inherent in free software, but there's so much duplication of effort out there, and sometimes it would make a lot more sense if developers worked together on a grand project rather than having countless half-baked apps and standards doing the rounds.

So if Flatpak were just yet another packaging format a la Deb or RPM, our shoulders would be aching from shrugging by now. But it's not. Flatpak promises to fix something that has been deeply wrong with GNU/Linux for many years: the almost unbreakable tie between packages and distribution releases.

Q But how is that a problem? I've been using \$DISTRO for years and always get the software I need!

Flatpak promises to fix something that has been wrong with GNU/Linux for many years: the tie between packages and distro releases

A That's true. If you're using a distro with large package repositories, such as Debian, Fedora, OpenSUSE and their derivatives, you've always had a wealth of software to choose from. But there's one major problem here: what happens when you want to install something that's not in your distro's repos? What happens if your distro only has FooApp 1.1, but the developers of FooApp have released 2.0 with a tramload of new features and goodies to play with?

Well, you're up a certain creek with a tiny paddle. Yes, maybe the FooApp developers have spent ages creating new packages for every major distro. Or perhaps there's some kind of backports repository where you can find it. Or maybe you're running a rolling-release distro, but they have their drawbacks as well (everything is a moving target). So what do you do? Most people who don't have the time or nous to get the new version will just wait for the next round of distro updates – even if it's another six months away.

Flatpak aims to fix this by making cross-distro packages possible. So regardless of what distro you're using, when FooApp 2.0 is released, you can install it straight away, without having to wait for your distro to package it up, or compile it from source, or do any other kind of technical gymnastics just to try out a new program.

Q Wait a sec – hasn't this been tried before? I remember hearing about Autopackage...

A Yes, there have been numerous attempts to do something like this. It's a tough nut to crack. But Flatpak is rapidly maturing, it has the backing of major distros, and the technical structure on which it sits is well thought-out. Autopackage used all manner of tricks and cludges to just about make a binary executable file work across multiple distros, regardless of the libraries installed, but it never really took off.

Flatpak, on the other hand, is ready to use today. There are already some big-name open source applications such as *LibreOffice* available in Flatpak format, and the range is growing. Flatpak was originally designed by Alexander Larsson of Red Hat, and while it's true that Red Hat is a major backer of the technology, developers

from many other projects and distros are involved as well. So we expect it to stick around for a while.

Q OK, this sounds pretty good. So how does it work?

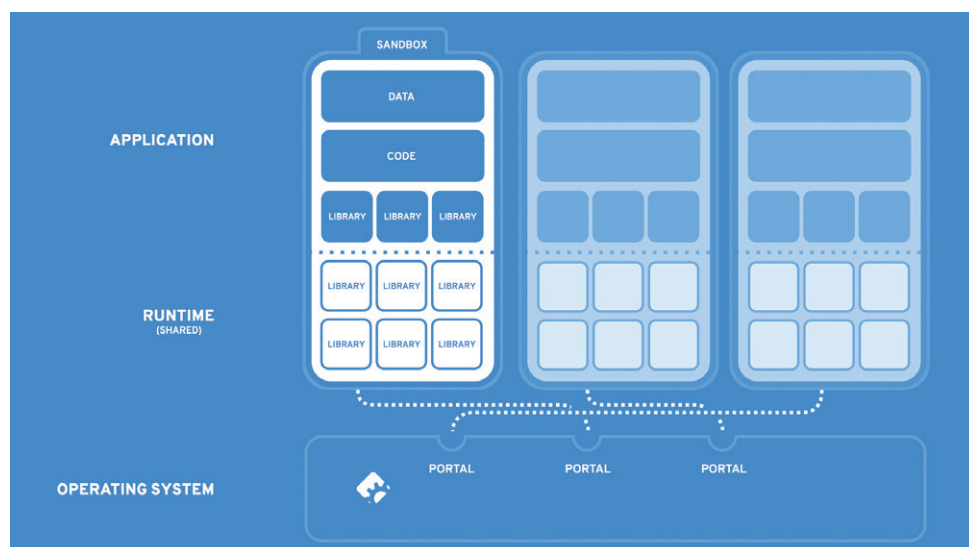
A Flatpak builds upon various technologies that are starting to emerge or become mature in the Linux world. For starters, it uses cgroups and Linux namespaces for “sandboxing” – that is, keeping each Flatpak program separate, so they can’t interfere with one another. The goal here is to ensure that a malicious Flatpak can’t do too much damage to the system, or mess with the workings of other software.

Now, you may think this is a bit paranoid, as Linux has barely been affected by malware so far in its life. But that’s because most of us install software from a vetted and well-maintained distribution repository, where we can be pretty much certain that the software hasn’t had dodgy backdoors injected. Repositories are not perfect, but they’ve done a great job over the years.

With Flatpak, it will be easier to download and install a random program from a random website. More users will start doing this, so it’s essential to avoid the problems that plague Windows when people double click every .exe file they see (or get sent in spam emails). So with Flatpak, applications are isolated from one another, and a malicious app is restricted in the harm it can do.

Q OK, but what about the dependency problem? Isn’t that the biggest issue of all with packaging on Linux right now?

A Yes, it is. Say you want to install FooApp 2.0 as mentioned before, but your package manager says you need **libfooobarbaz-12.0.1.3** whereas your distribution only provides **libfooobarbaz-12.0.1.2** (which turns out to be incompatible, because the developers didn’t follow a sensible version number scheme). So you decide to compile **libfooobarbaz-12.0.1.3** from source, install *GCC* and the whole toolchain kaboodle, find out that you also need to compile and install **liblolwut-0.2.5** and **libohreally-9.3.6** and then just get angry and close your laptop and go out for a walk.



Flatpak apps are sandboxed from one another, but use shared runtimes for dependencies.

What Flatpak does is this: it uses a system of “runtimes” which provide a set of base libraries that apps can depend on. These runtimes are the same across distros – so you don’t need to worry about minor version changes messing everything up. If you install a Flatpak app that uses Gnome libraries, Flatpak will first check whether you have the Gnome runtime installed (and if not, install it). There are various runtimes providing a reliable set of dependencies, and while they may eat up a chunk of disk space, especially if you have runtimes for multiple Gnome versions installed, it makes it easier for package maintainers to know what’s available in a distro.

Q And these runtimes include absolutely every library under the sun?

A Well, no – that’d be bonkers. There has to be a trade-off somewhere. Runtimes include major libraries like *GTK*, but if you want to build a Flatpak package that includes an obscure library not provided by any runtime, you should roll that library into the Flatpak itself. This keeps the Flatpak app nicely self-contained and means that users don’t need to hunt down extras.

There is, of course, a downside to this. In a normal Linux distribution, an obscure library – let’s call it **libhardlyused** – would be provided separately in the package repositories, and used by a handful of programs. If a security hole is discovered in

libhardlyused, the distro maintainers will patch it up, issue an updated version, and that small bunch of programs that use it will benefit from the update.

With Flatpak, each app that depends on **libhardlyused** will have its own version. So when a security vulnerability comes along, every app using **libhardlyused** will need to be updated separately. Some apps may receive updates quicker than others – some may not be updated at all. So the classical distro approach with its zillions of packages provides better security in some cases, but Flatpak strives for convenience as well.

Q OK, so I guess time will tell if it really takes off...

A Exactly. The pace of change in the Linux world is extremely rapid, and we love that, but it still takes a long time for new technologies to really propagate into mainstream distros. For how long have we been waiting for Wayland, for instance? (OK, there are still issues to fix, but it seems like it has been ready for the next round of distro releases for five years now.)

Still, you can try it for yourself today by visiting <http://flatpak.org> and following the installation instructions there. The website also includes a guide to making your own Flatpak, so if you’ve been working on your own Free Software project but found it a pain to package it up for multiple distros, give it a try – it may save you a lot of time in the long run. 🐧

“

Micromanaging people is not good in open source, so when they want to do something, they will do it – you just give them the frameworks and let them do that they want with them

”

FRIDRICH STRBA

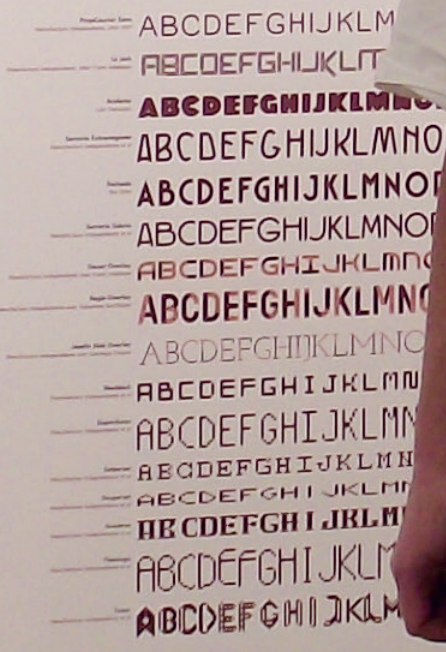
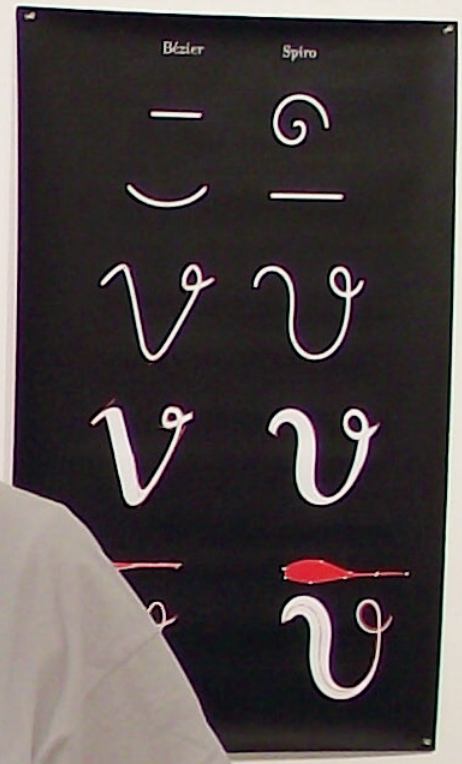
Ben Everard meets the man behind free access to file formats.

Data. Whatever you do with your computer, data is going to be at the heart of it, whether it's an office document, a chunk of program code, some images, or any other set of 1s and 0s. The important thing for this data is that you can access it from multiple programs so that if there's ever a problem with your chosen tool, you can still use the precious

information. Proprietary software isn't always designed with this information sharing in mind. After all, if only one piece of software can read your data, you have to keep using that bit of software – and buying updates.

For many years, the inability of open source software to consistently read and save data in the latest proprietary office

documents was a major stumbling block for Linux adoption, but now, the vast majority of open source office tools can read and write (almost) perfectly. This change is due to the hard work of a lot of people, and leading the charge has been the Document Liberation Project. We chat to Fridrich Strba, co-founder of the DLP, to find out what's been going on.



Can you tell me a little bit about how the document liberation project got started?

Fridrich Strba: Actually, it got started by the *LibreOffice* community. At that time, LibreOffice had good support for quite a lot of proprietary file formats, especially *Microsoft Office* formats, but then there were parts of *Microsoft Office*, like *Visio*, that were not supported. We were always trying to support it because people were asking for it. We realised that Valěk Filipov (who is now collaborating with me) found a way to reverse engineer the file format. Because the file format is binary compressed with a custom compression, it was not simple to understand how the files look inside, and he managed to do it.

This helped us to put it as a Google Summer Of Code project, so maybe some student would start to work on it, and we managed to get a student. By that time, I also had time not only to mentor the project but also to code on it – basically tandem coding with the person. After three months we managed to get something reasonably nice done, and it encouraged us. We're not extremely good communicators – we were communicating through code – so we got a method of work that was good for us. For example, if there was a commit

that was not working, instead of saying 'oh that commit doesn't work', the one who saw it fixed it and we advanced quite quickly from there.

From this one file format you decided to expand?

FS: We became emboldened by this result and we started to look towards other file formats. Valěk has quite a good experience of looking at files and trying to find patterns and such things so we managed to work with *Corel Draw* and then *Microsoft Publisher*, and then we were going on and on. At a certain moment, we realised that the best way to make it not just depend on us two was to create a project and make the code a little more modular so people can re-use the framework and let them do what they want to do.

Generally micromanaging people is not good in open source: when they want to do something, they will do it. You will just give them the frameworks and let them do that they want with them.

We gave it a legal foundation. It was kind of a constellation of libraries that was gravitating around each other. We built the project so that it has some existence beyond the libraries so that if one of the libraries has a bug and the person who works on it doesn't have much time, you still feel a bit obliged to

look at it. That was how it started, with the project itself and the legal framework. The collaboration started in 2011, and the project as a framework started two years ago – 2 April 2014.

You've supported several file formats over several versions. In general, are they getting easier or harder to work with?

FS: We have a theory about incremental reverse engineering, because even if there are several versions of a file format, no company can completely re-write their software. If you know a certain version, you can try to go lower because certain data structures may change, or maybe some representation of numbers may change, but nobody's going to completely change the file format because that takes several years to get

I don't think anybody wants to make it harder to read their files...

right. For example, with *Corel Draw*, we started with versions 7, 8 and 9, and then we went lower and we support everything from the beginning of *Corel Draw*. We then looked at the later versions... there were some little differences, but it's not like you have to completely reverse engineer a new file format. You can just look at the differences, and since we have tools that can show you binary diffs of chunks of the file, it was pretty easy. It was still a chunk of work, but it was not completely different.

Are there many new file formats that you want to be able to support?

FS: There are file formats that we'd like to support and we don't know really how to support them because the file format is really complicated, like for example *InDesign*.

We can support *Visio* files in everything that exists from *Visio 1* to the one that comes with the newest *Microsoft Office* – that's actually XML-based, but still the data structures are the same. With *Corel Draw* we support everything from version 1.



Do you want to help open up closed file formats? Pop by the #documentliberation-dev channel at irc.freenode.net and say hello.

Versions 1 and 2 are completely different – OK, not completely different, but the encapsulation was different. Now the last one we did was version 8, which was released this year – we realised that because of the way we parsed the documents, we were also able to parse the new documents with the old code.

LV What, in your view, is the toughest file format that you support?

FS: *InDesign* is complicated because it's basically a database, and it changes with each version. Even Adobe is unable to save files in lower versions – it only has migration from the lower version to the current version and then you can't save it in the lower version any more.

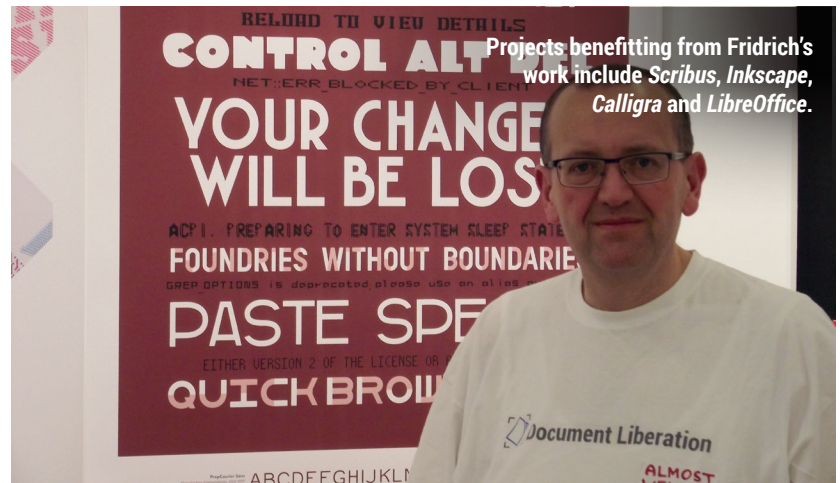
Freehand is quite a tough file format because you have to have all the records in order to be able to parse them, because they don't have size information. You have to know how to parse each record to be able to jump to the next one.

LV Have you ever come across anything that you feel has been put in deliberately to make it hard to reverse engineer?

FS: Frankly no. I think that people just try to dump their document in the files somehow... There are some custom compressions, but they didn't do it because they want to obfuscate it, but because they want to have it as a feature – considerably smaller files. I don't think anyone wants to make it harder to read the files.

LV For you, personally, what got you interested in this area?

FS: It's the technical challenge. I'd worked on other file importers that



were actually documented before, so I knew the frameworks we were using, and we evolved them into something self contained. I was working on something that imported *Word Perfect* files as my first open source project. It was not my project, but I contributed to the project, and I suddenly became maintainer when the other maintainers didn't have much time to do it.

Then I worked on *Word Perfect Graphic* file format, and I tried to unify the APIs to extract them from the libraries because the data structures could be the same. The callback functions weren't the same, but at a certain moment, we were at a point where the data structures were from the *Word Perfect* library, the interface for putting out the images was from the *Word Perfect Graphics* library, so if you wanted to import something from them, you had to have the *Word Perfect* library and the *Word Perfect Graphics* library. We decided that what can be common, we put into a single function and make the other things pluggable.

LV You mentioned that The Document Liberation Project came out of LibreOffice. Do you work

with other software as well?

FS: This application is used by *Inkscape*, it's used by *Scribus*, it's used by everybody who does something with importing the file format – it's used by *Calligra*. If they use these files formats, they use our libraries, because at a certain moment, we killed the market. It doesn't make sense for someone to re-invent the wheel in another framework because we tried to make the framework very independent – we don't have any *GTK* structures or *Qt* structures, we have just our structures that are good for the libraries. They're pure C++ and you can plug it easily, so there's no real reason to do it differently.

LV Are you looking for more people to get involved?

FS: Oh yes. It's always good for people to come. We need people! Well, what we don't need are talkers, but anybody who can make any meaningful contribution is welcome. If you want to import something, start to create your library and we'll integrate it in our framework. You can do that and we'll make you famous. We can't make you rich because we haven't made ourselves rich! LV

OLETOY

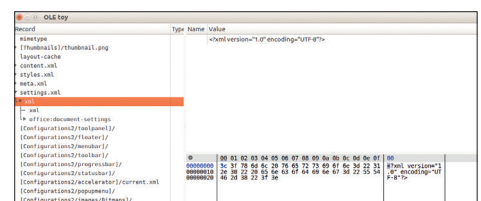
If you want to start poking around inside files to see what's going on, and potentially reverse engineer new file types, *Oletoy* is the tool of choice for the Document Liberation Project. As you would expect, it's Free Software, and you can download it from GitHub with:

git clone <https://github.com/renyxa/re-lab.git>

Then you can start the application with:

./re-lab/oletoy/view.py

This will start a graphical application that enables you to open files of many different types. You can see how the data's arranged in the file in text and hex format, and known structures will be broken down to make them easier to follow. In the Edit menu, you'll find options to manipulate the file by adding or removing data.



If you want to add support for a new file format, *Oletoy* can help you find out how it functions.

BUY LINUXVOICE MUGS AND T-SHIRTS!



shop.linuxvoice.com

REVIEWS

The latest software and hardware, rigorously bashed against a wall by our crack team.



Andrew Gregory

It turns out that yoghurt pots half-filled with Belgian trappist beer really do trap slugs.

In this issue's news roundup Mike reports that Linux has now reached a whopping 2.8% market share in desktop computer use. This is, quite simply, amazing. Viewed in isolation, 2.8% seems piffling, but for most of the last 10 years, the figure has been hovering around at just over 1%. I love statistics, so I'm choosing to interpret that roughly 1.8% percentage point rise as a 180% increase. Extrapolate that over the next 10 years and Linux will have achieved the utter domination of running 999% of all desktop computers.

Though mathematically implausible, I contend that this huge expansion will be made possible by the huge increase in the number of devices that we can reasonably describe as a desktop computer. Fridge? Baby monitor? Television? Smart insulin pump? Trip computer in your car? These are desktops. Kind of.

Silliness aside, 2.8% is huge: it's a growing, commercially relevant slice of the computer using market, and it means, finally, that Linux can't be ignored. Just remember that you liked Linux before it was cool.

andrew@linuxvoice.com

On test this issue . . .



Fedora 24 Workstation

The flagship Gnome distro gets more polish, more software and more excellent. Ben loves it.

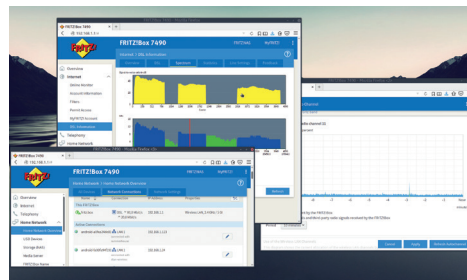
42



RetroPie

Mike enjoyed playing with this emulation setup so much that we haven't heard from him in weeks.

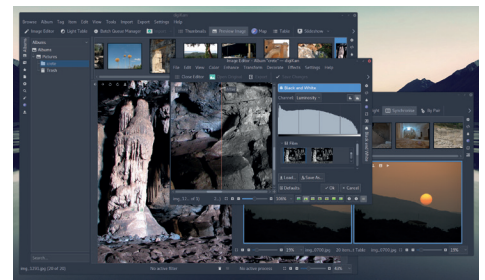
43



Fritzbox

It's a router, Jim, but not as you know it: this little device will make the master of your home network.

44



Digikam 5

Photographers of Linux, fire up your package managers and download this wonderful tool.

45

Group test and books



Boooooooooooooooooooooo!!!!

Penetration testing relies on a standard set of tools, but wouldn't it be awesome if you could craft your own in Python? Yes – yes it is awesome.

48



Group test – virtualisation platforms

When you're sick of the whirring, chugging sound of a hard drive being overwritten, it's time to virtualise your distro experimentation with one of these.

50



Fedora 24 Workstation

Ben Everard doffs his hat to a cutting-edge Linux distro.

Web <https://getfedora.org>
Developer Fedora Community
Licence Various free software licences

Fedora – the community Linux distribution sponsored by Red Hat – continues its endless march to the future with version 24. As always, this latest release comes jam-packed with the latest Linux technology. The big new feature in 24 is support for the Flatpak packaging format, which provides developers with a way of releasing their software in a controlled, sandboxed environment. See the FAQ on page 34 for more details.

Fedora defaults to the GNOME desktop environment and this release comes with version 3.20 (named Delhi), which brings with it a host of improvements. Our favourite addition is the new shortcuts window. Press Ctrl+? or Ctrl+F1 in any of the GNOME applications and you'll get a new window showing the shortcuts for that software. It's a simple idea, but

it makes it much easier to get to know the software. Other updates include an improvement to the search function in *Files*, which makes it easier to narrow down the results by type of file and date the file was used, and the addition of simple editing to the *Photos* image viewer.

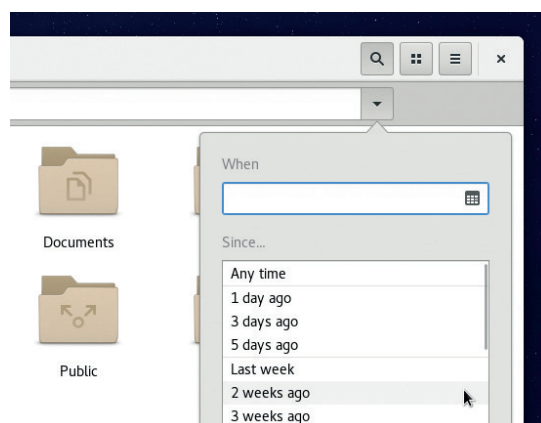
All about choice

If you'd rather a different desktop, there are spins for KDE Plasma, Xfce, LXDE, Mate, Cinnamon and Sugar (an environment designed for children).

It seems that with every Fedora release, we talk about Wayland, the next-generation display server. It's still not quite ready for prime time, so by default, version 24 of Fedora will stay with the older X server (adventurous users can install Wayland if they wish). At the time of writing, the Fedora team are planning to make the switch to Wayland with the next release (25), however, this change has been delayed so many times that we would caution against placing a bet on it.

As well as the Workstation version that we've reviewed here, there are server and cloud builds for running in other environments. Fedora 24 delivers exactly what we've come to expect from Fedora – the latest Linux tech bundled up in an easy-to-use format.

Flatpak and GNOME 3.20 make this a useful release even though Wayland is delayed.



The enhanced file search allows searching by date or file type without resorting to the terminal.



RetroPie 3.8

Got a Raspberry Pi sitting around doing nothing? Install this.

We dread to think of how much time we spent playing games on the classic 8-bit and 16-bit consoles and computers in the 80s and 90s. Actually, forget that – playing those games encouraged us to explore computers further, start writing rubbish little shoot-em-ups in BASIC, and finally move on to the black art of assembly language. Today, nothing fazes us, and we can hex-edit filenames in initrd images with both hands tied behind our backs.

RetroPie is a specialised Raspbian-based distro for the Raspberry Pi that focuses entirely on retro gaming. You write it to an SD card, connect your Pi to a telly, plug in a couple of USB joypads and *voilà*: you can emulate games released for the SNES, Mega Drive (aka Genesis), Game Boy, Game Gear, NES, Master System, PC and other platforms. Indeed, with a Pi 3 you can try emulating more high-spec consoles such as the PlayStation and N64, but performance and compatibility varies significantly. Stick with the 8-bit and 16-bit consoles and you'll be fine.

When we first started using RetroPie a couple of years ago, it was impressive but rather fiddly; you had to attach a USB keyboard, exit out of the shiny emulator front-end, and perform various tricks to get USB joypads properly configured in all of the emulators. This has been greatly simplified over time – now you just need to hold down certain buttons on

the joystick. Some advanced options still require command line fiddling, but for the most part it's a beautifully elegant plug-and-play retro gaming solution.

Even better, RetroPie lets you map a combination of buttons to exit the emulators, so if you have a few hundred Game Boy games installed on the SD card (which you obtained completely legally, of course) you easily can go through them, enjoy the good ones, and feel terribly sorry for the kids (or their parents) who splashed out £30 per pop on the rubbish ones. Note that if your USB joystick has shoulder buttons, you can use these to quickly skip pages in the list of games, which saves a huge amount of time when you have lots to try out.

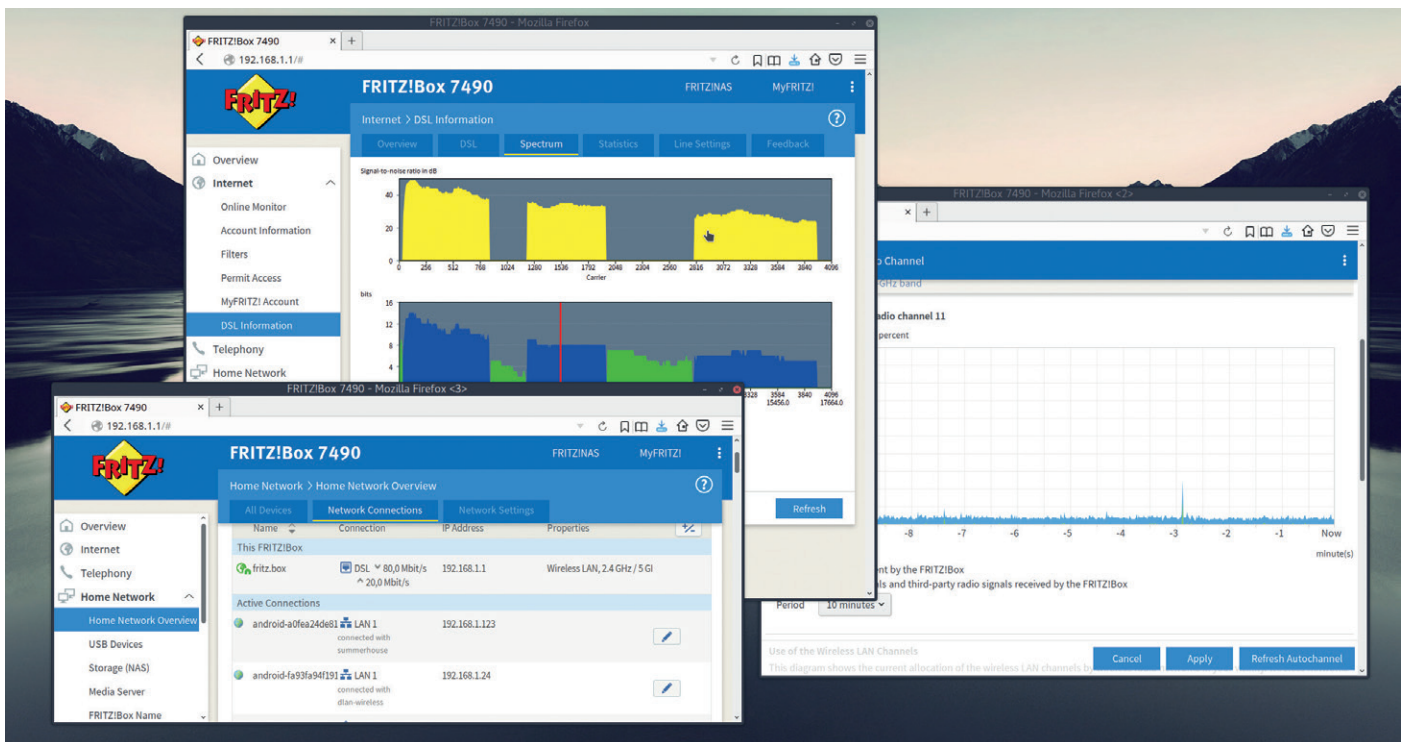
So if you have a Pi sitting around doing nothing special, and you want to see if you can beat your old *Super Mario Kart* lap times or complete *Streets of Rage II* without losing a single life, there's nothing better than this. 🎮

Retro gaming bliss – and even better that you can easily take it round to a mate's place.



If you're a Game Boy fan, you absolutely must play *Zelda: Link's Awakening*. No exceptions.

Web <https://retroPie.org.uk/>
download
Platforms Raspberry Pi
Price Free



FRITZ!Box 7490 Firmware 6.5

Graham Morrison avoids begging BT not to send him a HomeHub 6.

Web <https://en.avm.de>
Price £225

We reviewed the FRITZ!Box 7490 wireless router back in issue 10, and we liked it. In particular, it filled a difficult gap in the UK market for routers that can replace the generally woeful hardware you get from your internet service provider. More importantly, alongside ADSL, it also supported VDSL, which made it the only router we've found that worked with BT's consumer fibre network (BT Infinity).

The reason we're looking at the 7490 again is because there's been a major firmware update – the uninspiringly named 06.50 (now 06.52). This update changes almost everything about the devices, from a complete overhaul of the web interface and its functions, to the stability and speed of its wireless and internet connections. That last point was of particular interest to us as we had to stop using the old firmware when BT started to degrade the performance of our connection – a problem solved by going back

to the original router. One month and 247,057MB of downloads later, we've not experienced a single connection issue with the new firmware.

The Firm

The other major addition for this firmware is a completely new web interface. The new design is clearer and more responsive, and now works on small screens. This is quite an accomplishment for a piece of hardware that squeezes so many features into one place – there's USB network attached storage, media streaming, DECT telephony with fax and answer machine, home automation, guest Wi-Fi accounts, quality of service and per-device online filtering, to list just a few. All these functions are now easy to get to without any supporting apps, and work perfectly.

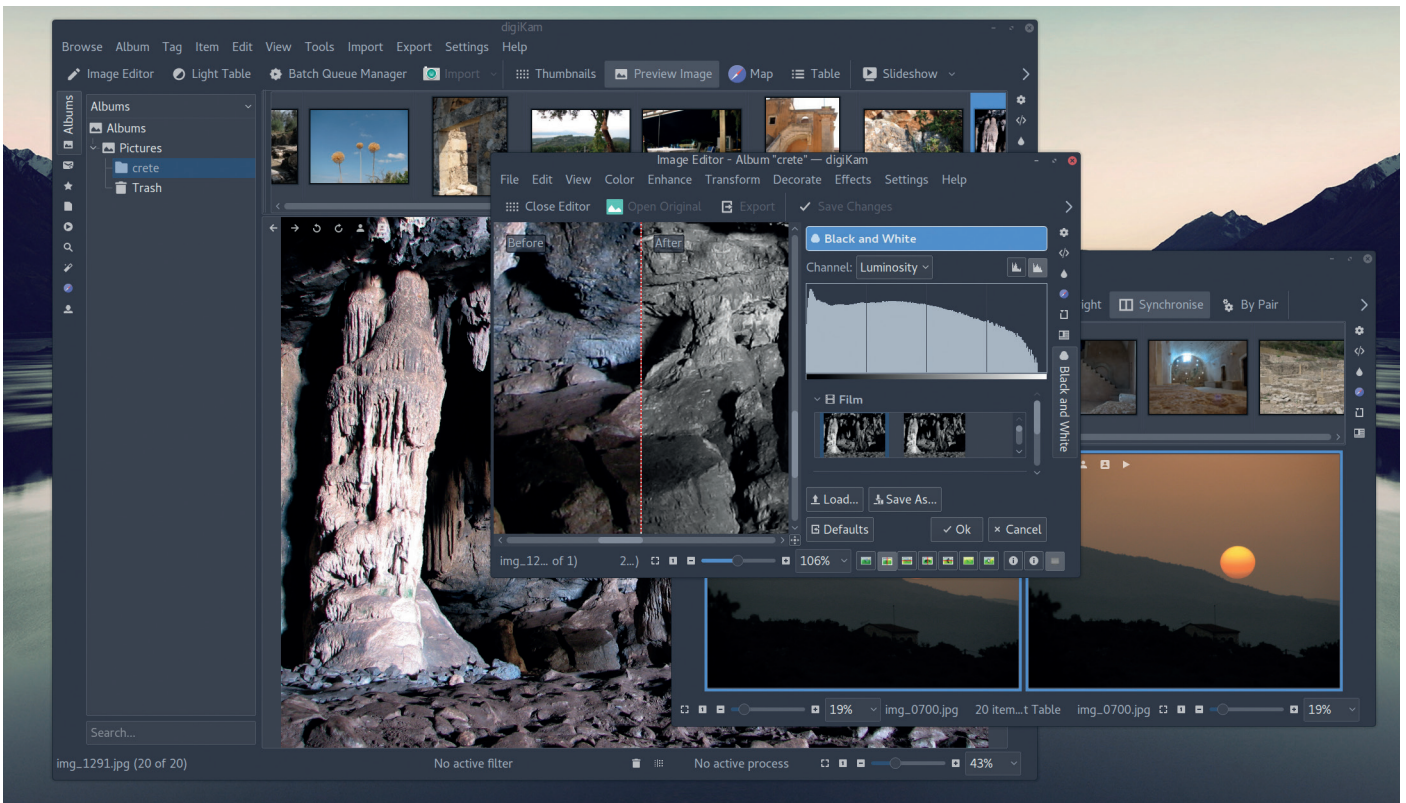
You get the same detail about your internet connection too, enabling you to change almost everything about how your network is configured. This is something you can't do with the usually locked-down routers provided by ISPs and gives the 7490 a clear advantage.

This is still an expensive router. But it combines the functionality of many devices, and the new firmware feels like a hardware upgrade to a device that was difficult to beat. 📺

Costly, but could replace several boxes with one. And the new firmware is like a new device.



The hardware hasn't changed, but the new firmware is so different from the old, it may as well be a new device.



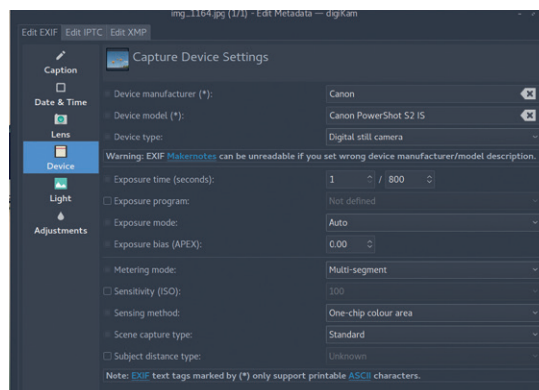
Digikam 5.0

Graham Morrison finally finds an app to replace his beloved Kalbum.

Digikam has always been brilliant, and has been our default photo management application for a decade. It handles all the formats we care about, including camera specific RAW images, and offers just the right balance of processing and editing for almost of the tasks we need to perform. Version 5 is a major update, and the result of two years' work by the developers.

Most of this work has been concentrated on rewriting almost the entire codebase to work with Qt 5. This was obviously a huge undertaking for an application as wedded to the KDE 4 and Qt 4 frameworks as *Digikam* was. Every KIO-slave instance was removed, for example, which required the database code to be replaced with a more platform-agnostic multi-threaded implementation. Like several other applications from the KDE 3/4 era, this means *Digikam* is now more portable, with both Windows and Mac versions being much easier to install, and there's a solid plan to remove the remaining dependencies to make *Digikam* almost dependency free.


But portability isn't the only reason to move to Qt 5: it's a much more modern platform, that's more efficient and more flexible than it's earlier revisions. It's the power behind KDE's Plasma desktop, for example, and helps to make *Digikam* better aligned with the Plasma desktop. Font rendering and theme integration is better, for instance, and despite the



Web <https://www.digikam.org>
Licence GPLv2+

One of *Digikam*'s best features is its metadata editor, where you can change almost any of the data held on your images.

complexity of the user interface, your large photo libraries, the editor windows and previews all appear more quickly than before.

The move to Qt 5 may not have added many new features, but that's probably a good thing – this is an application that does almost everything we want it to anyway. Whether it's playing with geolocation and the integrated maps, editing image metadata, merging bracketed images for HDR or the brilliant photo duplicate finder, you can't be serious about Linux and photography without adding this to your desktop. 

An excellent photography app, whether you're a beginner or a professional. Digikam is ace.



GAMING ON LINUX

The tastiest brain candy to relax those tired neurons



VULPINE CUNNING



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

Steam Machines were highly anticipated by the Linux community, but didn't manage to make the splash many had hoped for. Though they are far from dead, with vendors like Alienware releasing new models recently, it seems the strategy is a long-term one which gambles on the Vulkan API becoming the industry standard; Linux ports should thus become easier to the point where the platform can seriously compete with Windows in terms of titles. With seven of the 10 most popular games on Steam having Linux support, this doesn't seem like an impossible feat.

While the HTC/Valve Vive virtual reality headset has seen delays in coming to Linux, another VR headset known as the HDK2 by Razer already supports Linux and is due to hit shelves in July. While all the main VR systems are still works in progress since the technology is still in its early days, the HDK2 isn't just exciting due to Linux support, but also the \$399 RRP, which is \$200 and \$400 lower than the Oculus Rift and Vive respectively.

What makes this headset even more interesting is the use of the Open Source Virtual Reality (OSVR) ecosystem, an open standard for VR. It also supports SteamVR, so compatibility won't be an issue should OSVR not become the *de facto* industry standard. The 90Hz refresh rate and 2,160x1,200 resolution puts it on par with the mainstream VR specs, though some early reviews have found the visual quality lacking in comparison.

Hearts of Iron IV

Change the course of history.

Web <http://store.steampowered.com/app/394360>
Price £34.99

With *Hearts of Iron* now appearing on Linux with its latest installment, *Victoria* is the only one of Paradox's series of grand strategy games to be missing on Tux-powered machines. So if you've had a chance to delve into some dark ages or medieval strategy, or if those time periods aren't your thing, there's some World War Two to get into.

Hearts of Iron has typically strayed from the formula slightly, focusing more on warfare than nation-building, and the latest installment in the series is no exception. While there are political and technological aspects to explore, these mostly serve the purpose of ramping up mobilisation and improving weapons. While in other games it was possible to turn the likes of Ethiopia into a global empire, in *Hearts of Iron IV* this would result in a pretty boring game unless playing as one of the major Allied, Axis or Comintern nations. Though the possibility for some interesting alternative history scenarios exist with some secondary powers, such as winning the civil war as the Republicans in Spain, for the most part, not being part of the main conflict leaves little to do.



It is possible to play as any country existing in the time period.

On the positive side, the lack of an absurd amount of mechanics, and streamlining others like politics and trade, means that the game is far more approachable for newcomers than other series. That isn't to say that the game is suddenly less complicated than other strategy games like *Civilization* – the developers do recommend keeping the wiki open in a tab while playing the game, though it's still far less intimidating than *Europa Universalis IV*. In this sense, the game strikes a balance between satisfying the existing hardcore fans without others writing it off as over complicated. Overall though, it's hard not to feel like this is a step backwards in some regards, and *Crusader Kings II* and *Europa Universalis IV* are more immersive.



The game has a considerable focus on battle plans and the military-industrial complex.

Dead Island Definitive Edition

Back from the dead.

Web <http://store.steampowered.com/app/383150>
Price £11.99

Dead Island was released in 2011 and we got it on Linux in 2014, but now it's returned using the far more modern game engine used for its more mature cousin *Dying Light*.

The graphical overhaul is immediately obvious, and while not quite on par with *Dying Light*, visually it does look like something that could have been released recently. The original game wasn't without

issues on Linux, the worst of these being the inability to craft weapons effectively (one of the game's main mechanics); however these are gone from this version and performance is also decent with 60fps possible on mid-range hardware.

As far as zombie games go, it's still a decent enough game, but after playing *Dying Light*, it feels like a serious downgrade since there isn't really anything *Dead Island* does better. Still, at this price, it could be worth picking up for the extra dose of zombie killing.



The tropical island of Banoi and its undead inhabitants look better and deader than ever.

F1 2015

Great racing but no campaign.

Web <http://store.steampowered.com/app/286570>
Price £39.99

F1 2015 is the first game of the franchise to be ported to Linux and the latest racing game to be released. With such a major racing title coming out, we now finally have a decent roster of racing games on Linux after a considerable time with almost none.

F1 2015 does everything one would expect from a Formula 1 game, with the ability to choose from teams, tweak car settings and play practice races as well as qualifying and the main events. The physics and AI are solid, adding a great deal of challenge to the game.

There's also some impressive graphics and online multiplayer, however, it's the single-player campaign where the game falls short. Where previous games have



The graphics and racing are as good as it gets, adding far more realism.

featured a career mode, allowing the player to start in a smaller team, improve their skills and be offered bigger contracts, in *F1 2015* this is stripped down to the point where the single-player mode consists mostly of going through different seasons. The racing itself is among the most enjoyable out there, though those expecting a deeper campaign experience may be disappointed.

ALSO RELEASED...



Nation Red

This top-down zombie shooter harks back to an era where the player stood around and mowed down anything coming in their direction. *Nation Red* adds a lot to this classic formula, such as full 3D graphics and a decent variety of weapons and game modes. The ability to play something like this in online or local multiplayer makes it a lot of fun. <http://store.steampowered.com/app/39800>



Edna & Harvey: Harvey's New Eyes

This charming point-and-click adventure does extremely well with the genre's staple trappings. Its characters are very memorable, the humour is top-notch and the puzzles are entertaining, adding its own unique quirkiness and visual style that sets the game apart from the rest. The game's bizarre and imaginative world are tied up nicely with the story, making this a must-have for adventure fans. <http://store.steampowered.com/app/219910>



The Mean Greens – Plastic Warfare

Now ported to Linux, this third-person shooter where the player controls toy soldiers brings a breath of fresh air through an interesting aesthetic and lighthearted take on the shooter genre. There's a good variety of maps and game modes, while matches mostly take the form of five a side online multiplayer. Unfortunately, there's no longer a huge number of people online, but hopefully the Linux port should help remedy that. <http://store.steampowered.com/app/360940>

Maker's Guide To The Zombie Apocalypse.

When the apocalypse strikes, **Ben Everard's** obscure electronics hoard will pay off.

Author Simon Monk
Publisher No Starch Press
Price £16.50
ISBN 978-1593276676

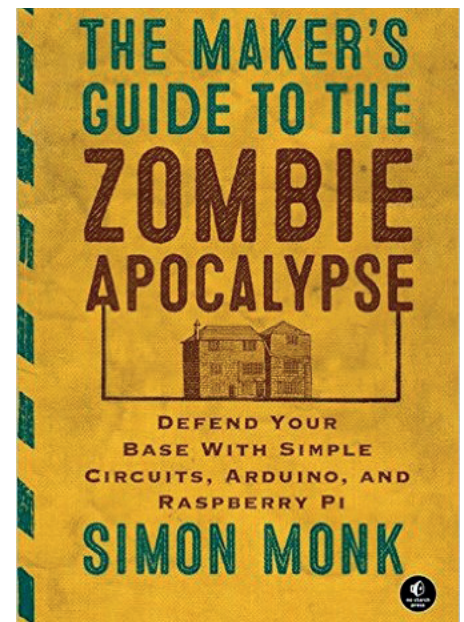
When society inevitably collapses into chaos and hordes of undead walk the earth, some people will be more equipped to survive than others. You don't need to stock up on supplies – everything you need will be available in abandoned shops – what you need is knowledge. Without the internet, there'll be no YouTube instructional videos or Wikipedia pages, so you'll only have what's in your head or on paper books.

The Maker's Guide To The Zombie Apocalypse tells you how to build the electronics you'll need in the post-apocalyptic world. You can learn how to generate electricity, and use this power to increase your chances of survival. If we live in fortunate times, and the zombie plague

doesn't strike during our lives, then it turns out that the skills you need to survive the zombie apocalypse are also useful if you're interested in building your own electronics.

The Maker's Guide To The Zombie Apocalypse is a good introduction to building devices with the Arduino and Raspberry Pi. You'll learn how to attach extra hardware, sense the environment and make the two devices communicate wirelessly – all great skills whether you're defending a base from zombie attack, making a smart home or taking your first forays into the world of robotics. Stay safe out there.

A fun and useful introduction to physical computing.



The information is useful in all post-apocalyptic scenarios, not just those caused by the undead.

Black Hat Python: Python Programming for Hackers and Pentesters

Ben Everard is slightly disappointed that this book isn't about snake millinery.

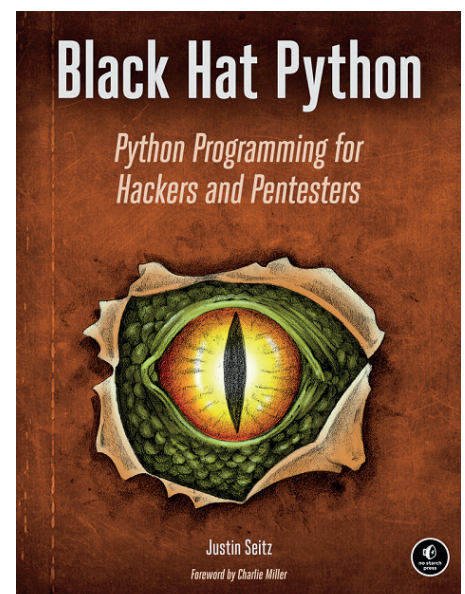
Author Justin Seitz
Publisher No Starch Press
Price £15.66
ISBN 978-1593275907

When it comes to ethical hacking, you can do a lot with the pre-made tools. The Metasploit Framework, the Burp Suite and others give you the capability to probe almost any desktop, sever or network without having to enter a line of code. However, at a certain point in your penetration testing, you will come up against the limits of these. Perhaps you're probing custom-made software that doesn't quite fall within the remit of the available tools; perhaps you're testing for an exotic type of bug; or maybe you just want to delve deeper and get a better understanding of what's going on – at that point, there's no option but to roll up your sleeves and write some code.

Black Hat Python helps people who understand Python and know the principals of penetration testing bring those skills together. You don't have to be an expert to get the most out of this book, but no time is spent helping the reader get up to speed.

The book guides the reader through networking, attacking websites, building a trojan horse, and then looks at some Windows weaknesses. At the end of the book, the reader should have a good idea what Python is capable of and how to use this language for software security. 📖

A flexible and powerful approach to penetration testing.



We strongly recommend that the information in this book is only used for white-hat hacking.

LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM



FREE SOFTWARE | FREE SPEECH

GROUP TEST

He might not be able to Alohomora his way through doors, but **Mayank Sharma** can whip up virtual machines out of thin air with little effort.

On test

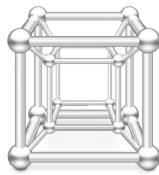
Gnome Boxes

URL <https://wiki.gnome.org/Apps/Boxes>

Licence GNU GPL v3

Latest release 3.20.2

Is simplicity the way to go?



Qemu

URL www.qemu.org

Licence GNU GPL v2

Latest release 2.6.0

Has the oldest app in the group test kept up with the times?



VirtualBox

URL www.virtualbox.org

Licence GPL and PUEL

Latest release 5.0.20

Does the app from Oracle deserve all the hoopla?



Virtual Machine Manager

URL www.virt-manager.org

Licence GNU GPL v2+

Latest release 1.3.2

Is Red Hat's challenger to Oracle any good?



VMware Player

URL www.vmware.com/products/player

Licence Freemium

Latest release 12.0.1

Is this freeware better than the free software options?



VMware Workstation

URL www.vmware.com/products/workstation

Licence Trialware

Latest release 12.1.1

Is it really worth all that money?



Desktop virtualisation apps

Truth be told, we've all got too much computing power at our disposal. It'll help you reduce the boot times and speed up application launches only up to a certain extent. One of the best uses of extra CPU cycles is to create virtual machines. This decades-old enterprise technology is now tame enough to be used by the average desktop user. The Gnome desktop environment even bundles one with the standard desktop apps such as *Gedit* and *Transmission*.

While they have clear advantages for enterprise deployments, virtual machines (or VMs) make a lot of sense on the desktop. You can use them as fully functional computing environments that are isolated from your main computing environment to test new software or even complete operating systems. If you don't feel brave enough to compile

a new piece of code on your main machine, tinker with it inside a VM. You can also use the VM to get a feel for FreeBSD, Haiku or some other esoteric operating system without jeopardising the partitions and contents of the disk on the computer. Furthermore, there are a couple of things you can do with VMs that you can't on a physical computer. For example, moving VMs from one computer to another takes a lot less effort than backing up and restoring a physical machine.

In this group test we'll examine some of the best apps that you can use to virtualise machines inside your home computer. Some have a very intuitive interface while others offer more features and flexibility. We'll highlight the strong suits of the individual apps and help you find one that best suits your requirements.

You can use a virtual machine as a fully functional environment isolated from your main computer

Virtualisation vs emulation

Virtualisation and emulation are two similar technologies that are often mistaken for each other despite several distinct differences. Emulation involves making one system imitate another. The most popular use of emulation is to run software that's designed for other hardware such as running console-based games on a PC. You can also use emulation software to conjure up complex pieces of hardware. For example, *Bochs* is an emulator that can emulate an entire processor in software.

The main difference between virtualisation and emulation is that while emulated environments require a software bridge to interact with the hardware, virtualisation accesses the host's hardware directly. Virtualisation involves simulating parts of a computer's hardware, but most operations still occur on the real hardware. Due to this reason, virtualisation is usually always faster than emulation. But unlike emulation, the host system has to have an architecture identical to the virtualised guest system.

Commonly used virtualisation jargon

Terms you should know.

Desktop virtualisation is just one aspect of the much broader virtualisation realm. While you don't need to have an in-depth knowledge to create and run VMs on your desktop, a familiarity with some of the most common terms will help you digest the trends and follow developments and news related to virtualisation with ease.

One of the most common terms you'll come across is hypervisor, which is the piece of software that enables you to create and run virtual machines.

There are several types of hypervisors. A bare-metal hypervisor, such as *XenServer*, runs directly on the hardware, and unlike hosted hypervisors like *VirtualBox* don't require a separate host operating system. Hypervisors rely on command-set extensions in your computer's processors to accelerate common virtualisation activities and boost performance. Intel-VT and AMD-V are the two sets of extensions for Intel and AMD processors respectively. There are other form of virtualisation besides full virtualisation,

namely para-virtualisation and operating system assisted virtualisation. These are used in server and large-scale environments and help minimise the overhead of running and managing a virtual environment.

Other terms you may come across include a snapshot, which is an image of the state of a VM at a specific point in time; and virtual appliance, which is a virtual machine with a fully preinstalled and preconfigured operating system.

Virtual Machine Manager

A very real overseer.

The kernel-based virtual machine (*KVM*) is the virtualisation infrastructure built directly into the Linux kernel, because of which it performs exceptionally well. Note however that *KVM* requires a processor with hardware virtualisation support. Fire up a terminal and enter

```
egrep -c '(svm|vmx)' /proc/cpuinfo
```

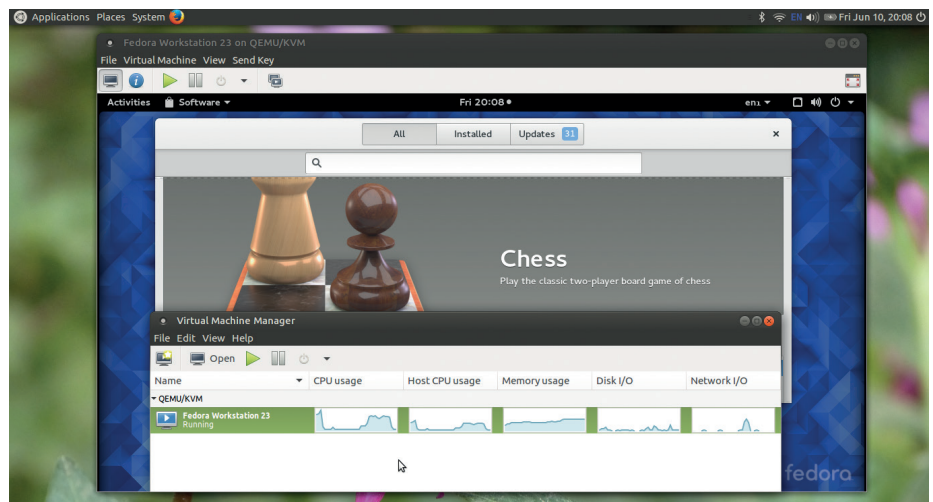
The command will return 1 or more if you have a processor compatible with *KVM*.

The Virtual Machine Manager, commonly referred to as *virt-manager*, is one of the most popular hypervisors, which interacts with *KVM* via the *libvirt* API to create and manage virtual machines (VMs).

Using the app is pretty straightforward. Like other virtualisation apps, *virt-manager* also employs a wizard to create new VMs. The five-step process begins with selecting an installation method. *Virt-manager* then asks you to assign memory, CPUs and storage to the VM. In the last screen you select the network settings for the VM and can also tweak other virtual hardware settings before powering on the VM. Using the app you can add and customise various kinds of hardware and controllers.

The app's main interface displays a list of all the VMs, and if one is running, it'll also display its live resource utilisation statistics. Inside the VM, sound works out of the box and the display can also scale to full-screen.

You can copy and paste text between the VM and the host, but can't move files in the same fashion. Also unlike other platform virtualisation tools such as *VirtualBox* and



Guests can use a couple of protocols to export their graphical framebuffers, including VNC and Spice.

VMware, there are no additional guest additions or similar extensions that will enable such features. *Virt-manager* can also attach USB devices including removable drives, webcams and Bluetooth devices found on the host and all work inside the guest seamlessly without any issues.

Senior management

Besides the usual slew of features, *virt-manager* also includes a host of functions that'll appeal to advanced users. For starters, while the default virtual hardware settings will work for most users, *virt-manager* offers fine-grained control over some pieces of hardware which will appeal to advanced users. For example, in the CPU section you can manually specify a CPU model for the guest. The section lists a huge number of CPU models such as Pentium 3, Opteron G5, Haswell, Westmere and more. If *virt-manager* isn't able to use the exact CPU model, *libvirt* automatically falls back to a closest model supported by the hypervisor while maintaining the list of CPU features.

The CPU section also lets you define the maximum number of CPUs accessed by the VM, which can be greater than the number of CPUs allocated by default. This enables you to hotplug additional CPUs as supported by *KVM* to cope with additional processing demands.

Also while *virt-manager* primarily creates *KVM* VMs, it can also manage *Xen* and *LXC* containers. Since *Virtuozzo 7* containers and VMs are managed via the *libvirt* API, it is possible to use *virt-manager* for the same purpose as well. Furthermore, *virt-manager* ships with a bunch of command line tools. There's *virt-install* for creating a VM, *virt-clone* for duplicating guests, and *virt-viewer* for displaying a minimal graphical console for the guest among others. One of the most popular is *virsh*, which is a CLI interactive shell for managing all aspects of a VM.

An intuitive and feature-rich app that works well for new users as well as the more demanding ones.



Gnome Boxes

The idiot's box?

Gnome Boxes uses *libvirt*, which also powers the *Virtual Machine Manager* and exposes just enough functionality to be usable while keeping it simple enough to make it approachable by virtualisation debutants.

It's no surprise then that the app has a simple user interface. When you wish to create a new VM, the app gives you a bunch of options. You can either enter the URL from which the app will fetch an ISO image and boot off it, or point to an ISO, which is the most commonly used option. If *Boxes* finds other *libvirt*-managed VMs on the computer it also gives you the option to import them. However this option is a work in progress and in our tests the imported machines fail to boot.

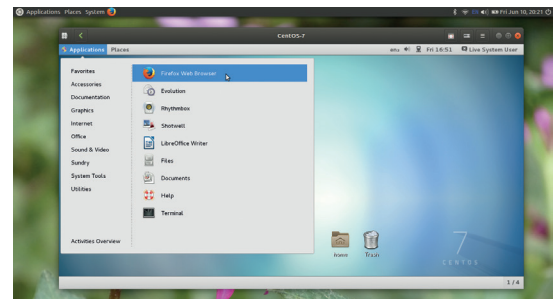
While *Boxes* claims to define the ideal settings for a VM after it recognises the ISO image you've pointing it to, the settings aren't as ideal as it claims them to be, and are very conservative.

Even when you want to manually customise the settings for the VM *Boxes* offers only two customisable hardware parameters, namely number of CPUs and amount of memory.

Simple stuff

Once the VM is up and running, *Boxes* behaves pretty decently. The VMs can switch to full-screen without issues and the sound works inside the VMs without any issues. You can also copy and paste text between the host and VM, and it has support for taking snapshots.

Like the other tools on test here, *Boxes* also has an option to connect the webcam, Bluetooth, fingerprint reader and other devices found on the host to the VM. When toggled, the devices disappear from the host and show up in the output of the `dmesg` command on the guest like they should. However, none of them work as they are



In addition to pointing it to an ISO image you can also point Boxes to a VNC server or oVirt and LibVirt brokers.

supposed to – the webcam throws input/output errors and the Bluetooth device isn't visible to other devices, for example. However, USB drives connected to the host show up without making a fuss.

The features are in line with its objective of simplicity, but this limits the app's usefulness in the long run.



VMware Player

Free as in cheap.

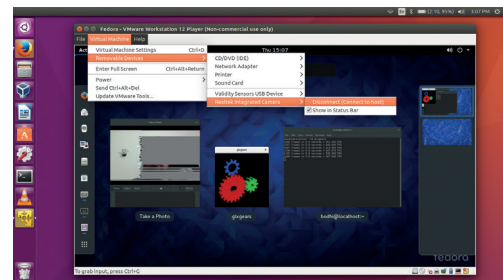
VMware Player is available as a free download for non-commercial use. According to its release notes, the latest version improves performance while suspending and resuming encrypted virtual machines and also support for 4K monitors. *Player* publishes a list of officially supported distros that it can recognise and for which it can configure appropriate VM settings. One of its unique features is the unattended Easy Installation mechanism, available for a few distros including Ubuntu.

Player's VM creation wizard is the standard affair and very easy to follow. You can also choose to customise the hardware at the end of the wizard before powering on the machine. *Player* can virtualise the usual slew of hardware such as disks, network adapters, sound cards and more. One interesting option is Printers, which enables the guest to print to any printer connected to the host without installing

any additional drivers in the VM. When you enable the VM printer, *VMware Player* configures a virtual serial port to communicate with the host printers.

Some of the useful features, such as the ability to move files and copy and paste text, requires the installation of the *VMware Tools* package. However, unlike with *VirtualBox*, installing *VMware Tools* is an antiquated process – you have to manually extract the tools and then install them via a text-based interface.

Some of the more interesting features provided by the add-on tools, such as dragging and dropping images between applications, work only between Windows hosts and guests. The Unity function is one of these – it enables you to run Windows from the guest on the host. However, like many of its unique functions, this isn't available for Linux guests and hosts. What's more, while *Player* can attach



You can access and download VMware's library of virtual appliances from within VMware Player.

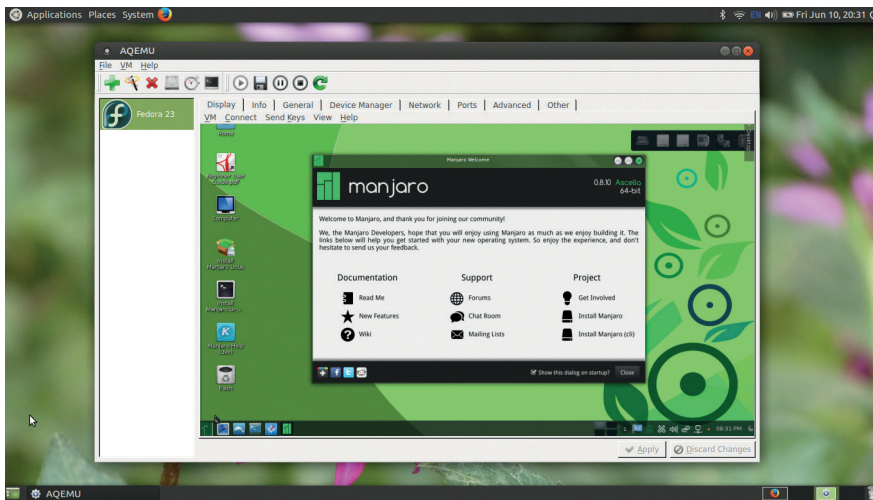
the host's integrated webcam, the image on the guest appears distorted. Another minor usability irritant is that shutting down a VM also exits the app. However, *Player's* biggest limitation is that it only lets you run one VM at a time, which is rather debilitating.

The free but proprietary app has a couple of interesting features that fail to make up for its lacklustre performance.



Qemu

Command and conquer.



Front-ends like *Qemu Launcher* and the recently forked *Aqemu* do a good job of providing a graphical interface to many of *Qemu's* advanced options.

Qemu is a very popular processor emulator and virtualiser that uses something called dynamic translation to speed up its magic. When used as a machine emulator, *Qemu* can run OSes and apps made for one machine, such as ARM, on a different machine, such as your x86 desktop. However, this dexterity comes at the price of performance.

To overcome this limitation, *Qemu* is often used together with the kernel's *KVM* module. But there's more to what you can do with *Qemu/KVM* than what's exposed by *VMM*. You can also run *Qemu* without a host kernel driver. When using *KVM*, *Qemu* can virtualise x86, server and embedded PowerPC, and S390 guests, while plain *Qemu* (without *KVM*) can virtualise architectures like ARM and PowerPC.

Qemu also boasts of impressive features. A couple of releases ago *Qemu* got a VirtIO-GPU driver for 2D graphics, which boosted the graphics performance of the guest machines. In the latest release the VirtIO-GPU driver even enables the guest systems to use the OpenGL acceleration provided by the host system.

The cost of *Qemu's* rich set of features comes at the cost of usability. *Qemu* is essentially a command-line utility and will typically install a huge subset of **qemu-** prefixed tools, each of which refers to a specific hardware architecture you can emulate with *Qemu*. While it's well documented and poses little trouble to experienced campaigners, grappling with

CLI utilities to create and define various aspects of the virtual machine isn't everyone's cup of tea.

For command-line lovers

Qemu supports various disk formats including **qcow2**, which is one of its most feature-rich formats. This format boasts of capabilities such as the ability to take multiple VM snapshots, AES encryption and zlib compression. You can present multiple virtual drives to the guest system by attaching up to four image files. Best of all you can also loopback mount a **qcow2** image on the host for transferring files between the guest and the host. You can also convert the image to the VDI format and use it with *VirtualBox*.

Qemu supports networking and can emulate some popular network cards. You can connect these virtual NICs to a *Qemu* VM using several different ways. The easiest of these is the user mode networking, which creates a private virtual network along with a firewall, a DHCP server, a DNS server and a Samba server.

Qemu also includes an interface for tasks like attaching USB disks and taking screenshots that you find with other apps. However, this interface is also CLI-based, unlike the ones found in other apps.

Boasts more functionality than the others, but requires familiarity with its command line tools.



A virtualisation distro

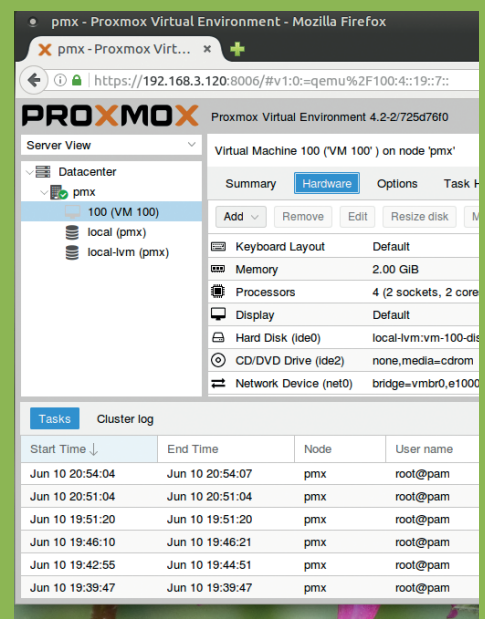
Create and manage VMs remotely.

Proxmox Virtual Environment (VE) is open source server virtualisation management software. Unlike apps such as *VirtualBox* and *Gnome Boxes*, Proxmox VE is a Debian-based Linux distro with a modified RHEL kernel. With the distro you can deploy and manage virtual machines. Proxmox VE offers the ability to manage both container-based virtualisation with *LXC* and full virtualisation with *KVM*.

The distro includes a simplified bare metal installer that takes over the entire disk. Once it's installed you can access Proxmox via a web interface from any computer on the network. The management interface includes a VNC console and supports SSL, and you can use it to create virtual servers as well as containers. For enhanced security, the interface supports multiple authentication methods and a role-based user and permissions management.

Proxmox offers several storage models. The virtual machine images can either be stored on one or several local storage types such as LVM and ZFS as well as on network shared storage like NFS and GlusterFS. Proxmox carries an integrated graphical backup tool called *vzdump*, which creates snapshots of virtual guests. The backup tool can do both scheduled backups and live backups and creates a tarball of the VM that includes the virtual disks and all the configuration data.

You can find lots of documentation and several video tutorials on the project's website. Proxmox is developed by Proxmox Server Solutions in Austria and is released under the Affero GNU General Public Licence.



You can download server appliance templates from within the Proxmox VE web interface as well as optimised appliances from the Turnkey Linux project.

VirtualBox vs VMware Workstation

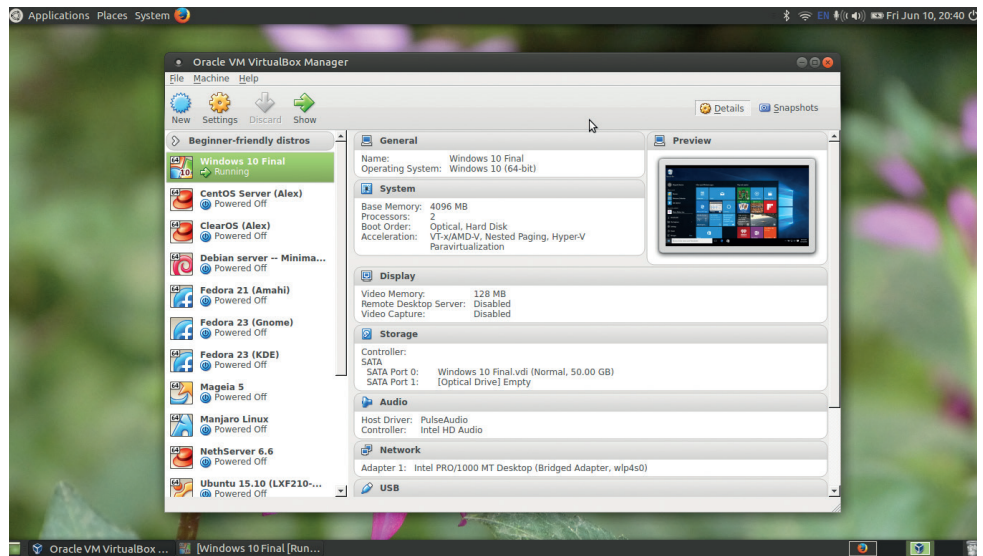
Feature-rich virtual machine builders.

The biggest difference between the two well-known platform virtualisation apps is cost. While *VirtualBox* costs naught, a single copy of *Workstation* costs \$250 (about £185) and comes with a complimentary 30-day installation support. Also *VirtualBox* is open source for the most part and is available in the repositories of the major desktop distros. However for some (albeit useful) functions *VirtualBox* requires the proprietary guest additions extensions.

VirtualBox can be credited for making virtualisation accessible to desktop users. The app offers para-virtualisation support, namely Hyper-V for Windows and *KVM* for Linux, which boosts the performance of the VM. *VirtualBox* also offers disk image encryption for improved security. However, this feature is only available if you install the proprietary *VirtualBox Extension Pack*. Some other features dependent on the proprietary add-on are support for USB 3.0 devices and bi-directional sharing.

Like other apps, *VirtualBox* takes you through a wizard to create a VM. Once you've created a VM you can power it on or tweak the settings for its virtual hardware. The VM settings window houses some useful options such as the ability to manually select a para-virtualisation interface for the VM. You can also mark virtual disks as hot-pluggable devices.

In addition to the desktop centric features, *VirtualBox* also includes several functions for advanced



You can run *VirtualBox* on a headless server and control it remotely either via third-party web-based interfaces or via its own extensive command-line tools.

virtualisation users. By default, VMs are isolated from the network. But if you're running a server inside a VM, *VirtualBox* can set up port forwarding to make sure the server is reachable from outside the VM.

Man your stations

VMware Workstation looks different from its freely available sibling tested earlier. For one, *Workstation* lets you run multiple VMs concurrently inside separate tabs. Its UI also presents additional options such as an interface to convert VMs into the OVF format and another to mount virtual disks on to the host. The wizard for creating a new VM

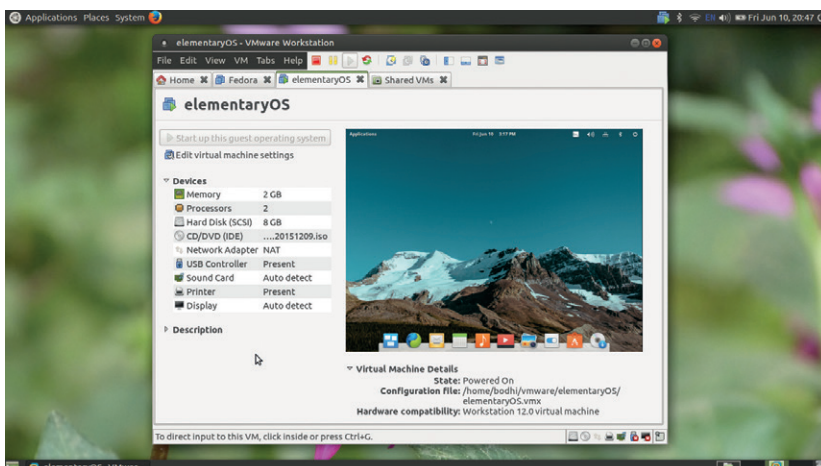
is the same as in *Player*, but with many more options to tweak the virtual hardware. You can, for example, define the number of cores for each processor you add to the VM. *Workstation* also includes the *Virtual Network Editor* tool that lets you create complex network configurations. The app also lets you encrypt VMs and restrict unauthorised users from modifying the VM.

Workstation's latest version supports DirectX 10 and OpenGL 3.3, which means it trumps the other apps in terms of rendering 3D apps and games. Unlike the other apps on test here, *Workstation* lets you allocate up to 2GB of video memory to a VM. *Workstation* also includes a command-line tool for operating VMs from the CLI, but it's not as extensive as *VirtualBox*. Both also have useful snapshot and cloning features to preserve the state of a VM and to duplicate a VM, respectively.

VirtualBox
Ships with enough features to satisfy both the desktop user as well as the advanced virtualisation campaigner.



VMware Workstation Pro
The expensive licence gets you features that make more sense to an enterprise user than on an everyday desktop.



You can run *Workstation* as a server to share virtual machines with others.

OUR VERDICT

Desktop virtualisation tools

Unlike some of the other group tests, this one was surprisingly easy to judge. Our unending love for open source software has made us intolerant of the tiniest of mistakes in proprietary software and for good reason. Why would you want to throw away money or your freedom over software that's inferior to free and open source options? It's because of these reasons that both of *VMware's* contenders lose out. The free-of-cost *Player* product loses out for failing to give us a compelling reason to recommend it over other options. Its biggest turn off is the inability to run multiple VMs at the same time. Its big brother, *Workstation*, fails to justify its cost for the virtualisation needs of the average desktop user.

We can also strike off the venerable *Qemu* from the list of contenders because of its basic interface. The comprehensive and robust CLI-based app involves a learning curve that's a bit too much for a desktop user pampered by graphical interfaces. The third-party front-ends do a commendable job of exposing some of its impressive capabilities, but many have failed to keep pace with the development of

Qemu. *Aqemu* has recently been forked and bears a new look, but you'll have to manually compile it.

Next we eliminate *Gnome Boxes*. It's a nifty little app but the biggest problem is that it is focused primarily on simplicity. That's not usually a bad thing, especially when rating apps for the desktop, but in the case of *Boxes* the modesty comes at the expense of several useful features.

Proprietary killer

The runners-up spot goes to *VirtualBox*. The recently released major version is a watershed release for the app which has managed to claw back onto the radars of serious virtualisation users after incorporating para-virtualisation abilities. However, to enjoy all its capabilities you'll have to rely on the proprietary guest editions.


Why would you want to use proprietary software when you can get the same function with FOSS? *Virtual Machine Manager*, our group test winner, comes equipped with several features that you get with *VirtualBox's* proprietary add-ons. *Virt-manager* is pretty intuitive to use and scales well, which makes it ideal for a large demographic.

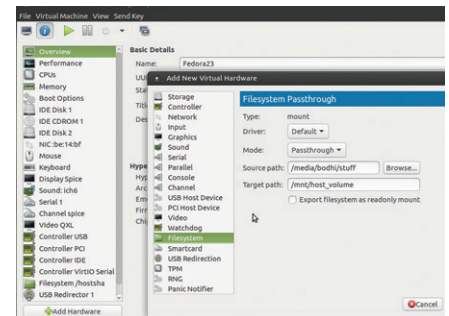
Why would you want to throw away money or your freedom over software that's inferior to FOSS options?

Virtualised servers

It seems ludicrous to put a server inside a virtual machine – a mission-critical server task requires a dedicated machine for reliability reasons, right? But those of you who only need a server occasionally can host them inside a VM instead of earmarking a physical machine for the task. For example, you can host an instance of your favourite web server to test code or even host websites for a limited audience by installing lightweight web servers inside the VM.

There are several advantages of this approach. First up, in case your server is exploited in some way, the damage the

attacker can cause is only limited to the virtual environment and cannot permeate to the underlying physical machine. Also virtual machines are more malleable and portable than physical servers. You can give it more storage space or RAM or even number-crunching prowess with only a couple of clicks. It's also easier to export a virtual hard disk and move it to another physical machine. Several hypervisors also enable you to pause your servers and even take a snapshot of a healthy working state of the server that you can revert to in the event of a mishap. 



Virtual Machine Manager lets you add a filesystem passthrough to share files between the guest and the host.

1 Virtual Machine Manager

Killer feature Rich set of virtual hardware customisations.

URL <https://virt-manager.org>

An open source app that'll satiate the virtualisation needs of a large number of users.

2 VirtualBox

Killer feature Intuitive interface.

URL www.virtualbox.org

One of the most popular desktop virtualisation app, which deserves all the accolades it gets.

3 Gnome Boxes

Killer feature Straightforward interface.

URL <https://wiki.gnome.org/Apps/Boxes>

This no-nonsense app is a wonderful starting point for virtualisation newbies.

4 Qemu

Killer feature The unfathomable list of customisations.

URL www.qemu.org

The very powerful CLI tool that can be of use to desktop users only via one of its graphical front-ends.

5 VMware Player

Killer feature Available free of cost.

URL www.vmware.com/in/products/player

The proprietary freeware doesn't really offer any compelling reasons over the open source alternatives.

6 VMware Workstation

Killer feature Graphics support

URL www.vmware.com/in/products/workstation



It's proprietary, expensive and is really designed for the enterprise and power desktop user.

Subscribe

shop.linuxvoice.com



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

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

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

Get 96 pages of tutorials, features, interviews and reviews every month

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

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

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

NEXT MONTH IN LINUX VOICE

ON SALE
THURSDAY
25 AUGUST



fedora™

FEDORA: IT'S AWESOME

Freedom, stability, a sense of community and whole load of cutting-edge software – find out what makes Fedora so darned good.

EVEN MORE AWESOME!



RetroPie

We've finally got it working, so we thought we'd share the most epic way on earth to play the games of Mike's youth on negligibly cheap hardware.



Ubuntu Snap

A new packaging format hoves into view, bearing a bushel of promises about dependencies, security and convenience. Were the prophets right?



Linux.gov

Her Majesty's government is switching some of its staff from *Microsoft Office* to Google Docs. Only LibreOffice can save us now...

LINUX VOICE IS BROUGHT TO YOU BY

Editor Ben Everard
ben@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Editor in hiding Graham Morrison
graham@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, 2nd Floor, 5 Churchill Place, Canary Wharf, London, E14 5HU
Tel: +44 (0) 20 3148 3300

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

Copyright Linux is a trademark of Linus Torvalds, and is used with permission. Anything in this magazine may not be reproduced without permission of the editor, until February 2017 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2016
ISSN 2054-3778

Subscribe: shop.linuxvoice.com

FOSSpicks

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



Our benevolent editorial overlord **Graham Morrison** tears himself away from updating Arch Linux to search for the best new free software.

Audio effects

Linux Studio Plugins Project 1.0.8

As regular readers will know, we really enjoy using audio and music software with Linux. Individual programmers and small teams have created unique software that can help musicians differentiate themselves from the huge mainstream of *Ableton Live* virtual DJs with their default set of presets and popular plugins. There's open source software for creating notes from algorithms, open source software for constructing your own sound generators and sequencers, and a thousand different effects to make your music sound like infinite variations of an alien landscape.

But regardless of whether you're creating pop music or musique concrète, you still need a core of effects and processors for day-to-day editing. These kinds of effects – the audio equivalent of

adjusting brightness or saturation in an image, have been few and far between in the world of open source, especially when you need quality output. But we're happy to report that this is a gap that the *Linux Studio Plugins Project (LSP)* fits brilliantly.

Plug and play

LSP is a collection of audio plugins that have been developed for processing audio. They can be used creatively, of course, but this is a collection of plugins that acts mostly as a toolkit. The entire suite is brilliant at fixing problems and making your audio sound better. Take the delay compensator, for example. This will delay the audio by a set period of time, and is essential if you're recording something with several microphones at different distances.

The microphones further away will obviously get the audio slightly later, and this can cause phase problems on playback as the offset energy in one waveform cancels out or emphasises the energy in another. Using the delay compensator, you can make sure every waveform is in synchronisation, as if each mic had been exactly the same distance from the source.

The plugin even offers 'distance' as a scale for the delay, so you don't need to perform calculations with the speed of sound on the back of an envelope. But it also includes milliseconds and samples, which is perfect if you want to compensate for a slow DA converter or piece of external equipment, or even the various delays introduced by software synthesizers and effects.

The most surprising inclusion is a very capable sampler. These are essential for simple sound triggering, both creatively as a sound source, and as a general production resource. They're great for backing tracks, for instance, or sound effects during podcast recording. The options won't match a professional sampler, but they're perfect for triggering sounds and loops. And the latest release of the plugin suite includes a trigger sampler that will play back sounds when it hears a specific frequency, rather than waiting for some MIDI input. This is a unique plugin, and one well worth investigating, as is the entire suite.



- 1 Sampler** Play up to eight sounds with your MIDI keyboard, or trigger effects and loops from a sequencer.
- 2 Delay compensator** Essential when recording multiple inputs with multiple distances.
- 3 Spectrum analyser** See audio frequencies you can't hear.
- 4 Stereo/Mono** Each effect is supplied as both stereo and mono version.
- 5 Phase detector** Eliminate that weird sweeping sound from your recordings.
- 6 VST and LADSPA** Native Linux VST, LADSPA and LV2 versions of the plugins can be installed.

Project website
<http://lsp-plugin.in>

Movie utility

Govie

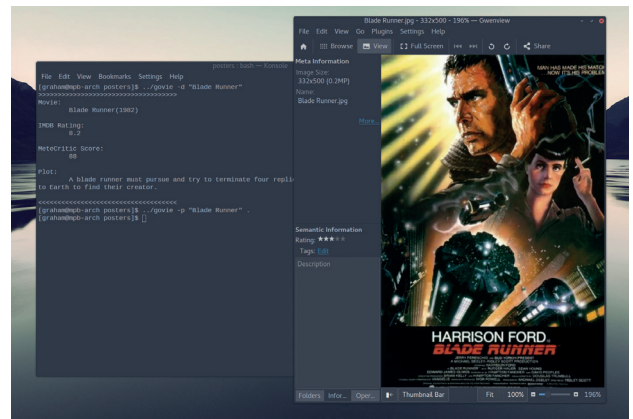
Until *XBMC* and *Kodi* made integrating film metadata into an open source application look easy, movie fans were often left to random searching and saved files from IMDB (the Internet Movie Database) to get their fix of movie memorabilia. If only they'd had access to *Govie*!

Govie is a very simple command line tool that queries the IMDB servers and delivers results directly to your command prompt. Typing `govie -d "Blade Runner"`, for instance, will return the year the movie was made, the IMDB review score for the movie, the MetaCritic score and a sentence or two on the plot, all delineated by 'greater-than' and 'less-than' symbols. If there's some ambiguity about the film you're looking for, you can search for a film in a specific year with the `-y` argument. The command is

obviously useful on its own because film fans can avoid loading up a browser if they need some information, but it's also possible to load up the IMDB page in your default browser with the `-o` option.

It's the scripting potential of *Govie* that we really like, as this command makes it easy to populate your own scripts and applications with the latest data from IMDB. In particular, the `-p` option will download a poster image for the movie you're searching for. You can even list more than one film in a single command, letting you quickly automate a collection of posters for any collection of movies you own or

Govie queries the IMDB servers and delivers results to your command prompt



Watch *Blade Runner* now, before Ridley Scott ruins it with a nonsensical sequel.

are interested in. We found this useful when filling in missing images from *Kodi*, and we could easily automate the download of posters for movies we record off digital television in the same way we used to for CD/LP album covers.

Project website
<https://github.com/narenaryan/Govie>

Theme editor

Plasma Theme Explorer

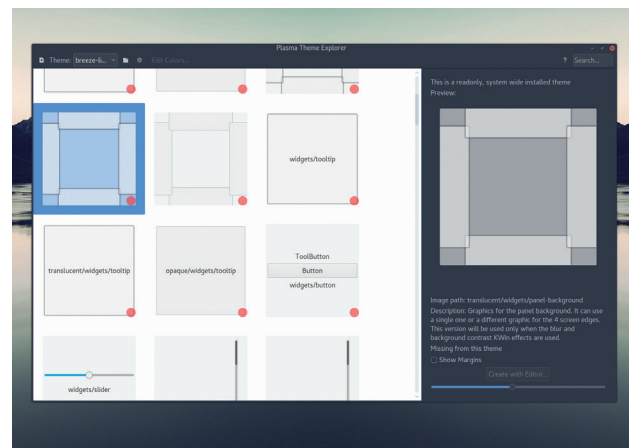
KDE's Plasma themes have become difficult to understand. In old versions of KDE, you used to be able to change many of the parameters that were responsible for how a theme looked, such as the amount of drop-shadow, or adjusting the blue glow that the desktop defaulted to for windows. The modern desktop eschews these options, and this is because the way themes are created has changed. Most themes are now a collection of scalable 'SVG' image files, slotted into pre-defined pieces of the user-interface like a jigsaw.

You need an image for the top-left corner of a window, for instance, or for the middle part of the desktop panel. Many effects, such as any drop-shadow or glow, are baked into these scalable

images, requiring them to be edited in an application like *Inkscape* if you want them changed. The main problem is that none of this is obvious, and exploring the various files is difficult. Which is exactly where *Plasma Theme Explorer* can help. It's been around for a while, and it might be difficult to find a package for your distro (it's part of Arch's Plasma packages, however).

When installed, it previews and lists all the graphical elements for a theme, so you can see exactly what they each do. If the theme is locally installed, rather than installed 'system wide', it also lets you open

Plasma Theme Explorer is a great starting place for creating your own themes



Dive into the details of how a Plasma theme is constructed, and change the parts you don't like.

these elements in an editor or adjust the colour palette. It's a great starting place for creating your own themes, or adjusting elements of a theme you've always wanted to change but never found the option.

Project website
www.kde.org

Desktop theme

Arc Dark

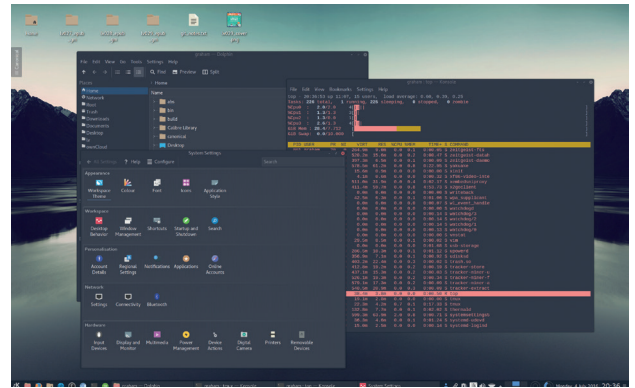
Despite many studies showing that black text on a white background is the best for readability, as this magazine proves, many developers seem to spend days in front of various terminals with a reversed colour palette – bright text on a dark background. This is because the brightness that surrounds the text can end up being more fatiguing on your eyes than the reversed text, even when the clarity of the text is slightly reduced or leaves shadows on your retinas.

This has spawned a plethora of terminal, *Vim* and desktop themes that attempt to better integrate reversed palettes into an entire working environment. Arc Dark is one such theme, and it's the best we've found, especially for GTK-based desktops and KDE Plasma. What we like most about this

particular theme is its breadth and consistency. It includes separate theme files for *Chromium*, *Firefox*, the desktop, the command line and even the *Kicker* app launcher and *Yakuake* drop-down terminal. The colours are dark but not black, reducing the contrast.

The only slight modification we needed to make was brightening the pale blue used for text, but we really like the palette generally. On a terminal, it's a cross between a dark solarised theme and Ubuntu's Tango, and because the same colour is used for both a window's background and for the window decoration, such as the titlebar, Arc

Arc Dark feels modern and integrated... a dark version of Google's Material design



Dark themes don't always look great in screenshots, but they can help tired eyes that spend all day in front of a screen.

Dark feels very modern and integrated, much like a dark version of Google's material design. The only element missing is a complementary icon theme, but we've found the dark icons from the 'Papirus' set integrate perfectly with the style and aesthetic of Arc Dark.

Project website
<https://github.com/horst3180/arc-theme>

Programming

Chuck

It's been possible to write code that's interpreted live and generates music for a while, and there are some electronic musicians that build an entire performance out of this method. Instead of a DJ with a real or virtual terminal, you can watch *Emacs* projected onto a large screen as interweaving beats and melodies are coded in real time. Lots of different languages are capable of doing this, but a language called 'Chuck' is the best we've found.

Chuck has been written specifically for audio and music projects. It can talk to MIDI and OSC synthesizers and audio equipment without any further libraries, incorporating all kinds of input protocols for things like data gloves, laser harps and even iPads. The latest version talks to Arduinos and

similar circuit boards, making it the perfect platform for music experiments. But most importantly, and because Chuck has been built for audio and music, timing is part of its fabric. Create two functions that generate a beat, for example, and run them both at the same time – Chuck will ensure they both stay exactly in time over hours, with sample accuracy, using a forking mechanic that Chuck calls a 'spork'.

You can spork many different processes all generating audio and note data, and they'll all keep exact timing with one another. You can forget about the complexity of threads, or of generating processes and signals that stay in synchronisation. You can simply get on with the job of creating sounds, whether that's at a sample level for something like frequency



This is the Chuck equivalent of "Hello World!", the code required to generate a sine wave.

modulation synthesis, or at a note level for Steve Reich-like interrelated fragments of repeating melody. Chuck accomplishes this with an easy to use JavaScript-like syntax that works brilliantly with Linux using both ALSA and PulseAudio – or no audio at all if you're after that John Cage sound.

Project website
<http://chuck.cs.princeton.edu>

Document conversion

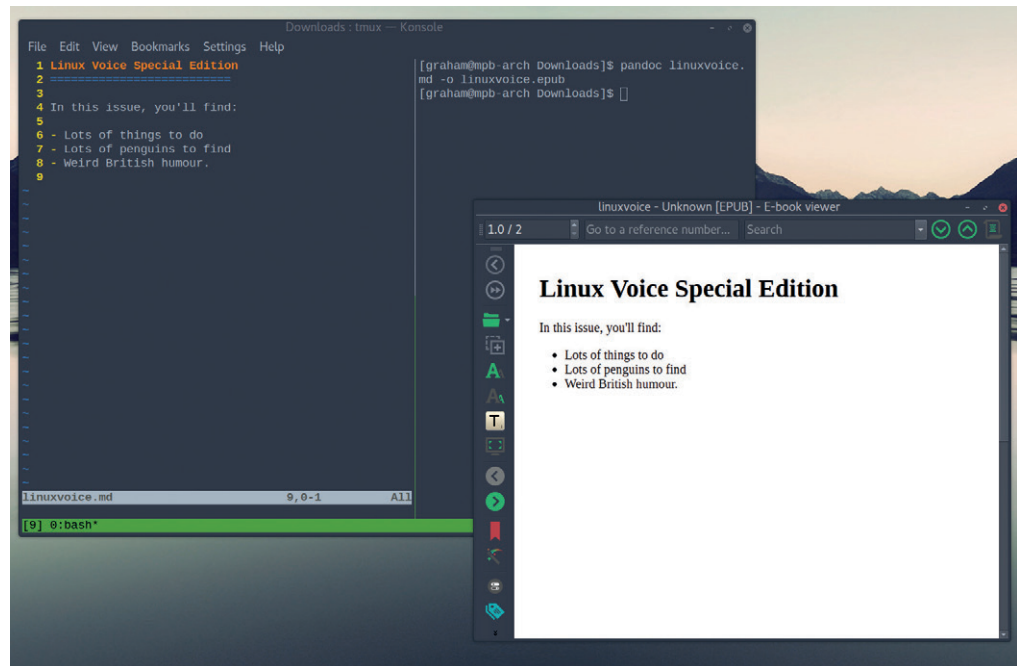
Pandoc 1.17.0

Pandoc has been around for a decade, but if you've never taken a look at this brilliant document conversion tool, now is a great time to do so. It's a command line utility that deals in the dark art of converting a document from one type to another.

Recently the source format of choice has come to be Markdown. Markdown is a simple way of marking text files to signal which parts are titles and subheadings, for instance, or items in a list or table (along with other markings). It does this in a way that doesn't break the readability of the original text file, marking a heading by placing a line of `====` symbols beneath, a sub heading with `----` or an item within a list with a preceding `-`.

When you've learnt this simple syntax, you can write in Markdown as quickly as with plain text. But because Markdown started off as an informal idea, rather than a corporate-sponsored specification, lots of different people have been making lots of small changes.

And this is where we've found *Pandoc* to be brilliant, because while it can be used to convert documents between re-usable text and word processors, it's best when it's converting between the broad churches of Markdown, as well as letting you dive into lots of specific arguments for each element you'd like converted and how you'd like them to appear in the end format.



The only problem with all this power is that *Pandoc's* array of options can be bewildering, and you need to have a good idea about the specifics of both your source document and your potential destination document to be able to choose the best command line arguments. Even then, we've nearly always needed to resort to trial and error. At its simplest, you call the command and tell it which formats you want to convert between:

```
pandoc --from=rst --to=markdown  
--output markdown.md input.rst
```

The power comes from being able to augment these simple commands with your own

The pandoc command can turn one document into another, but it's most useful for converting Markdown into almost anything else.

requirements. You can automatically word wrap, for example, by adding `--column=80`. You can generate headers that are compatible with GitHub's markdown by adding `--atx-headers`. There's even a specific module for GitHub-flavoured Markdown if you need an all-in-one solution, and getting documentation from source code into GitHub is one of *Pandoc's* greatest features.

Output modules, such as the Markdown one we're using in this example, can have features turned on and off, or strung together with a series of `+` symbols. You can define pipes as the separation character in table, and define backticked code blocks by following `markdown` with `+backtick_code_blocks+pipe_tables`, for instance, letting you carve your perfect output in a perfectly reproducible command, and that's without even looking at the ePub, Latex, Docbook and Docx formats that are also supported by *Pandoc*.

There's also a website that will perform the same actions on your documents without installing anything.

Project website
<http://pandoc.org>



Unix utility belt

moreutils 0.59

The tagline for **moreutils** is “a growing collection of the Unix tools that nobody thought to write 30 years ago,” and we can’t do a better job at summarising this excellent collection of tiny tools that do one job each, but do that one job well.

There are currently 15 commands in the package, including the dubiously named **pee** and the immensely useful **sponge**. **Sponge** is a great example of the kind of problems that these tools solve. It simply ‘soaks’ up the standard input and places it in a file. It will attempt to update a file, rather than replace one, and will wait for all of the input before writing. This makes it a good tool for pushing parameters into configuration files.

Other commands are more specific. **errno**, for example, will look up the error number names and

descriptions; **vipe** inserts a text editor into a pipe; **lckdo** runs a program that already has a lock; and **ifdata** is for grabbing a load of information on a network interface without going into **ifconfig**.

Let a thousand flowers bloom

We also really liked the tiny but immensely useful **ts** command. When running, this inserts a time stamp of anything pushed into its input, whether that’s the time of a line in a script being executed or the time some debugging output was delivered. The project has become so successful that there’s now a queue of other small but general

“...a collection of the Unix tools that no-one thought to write 30 years ago”

This is a collection of tools that adhere to the old Unix philosophy of only doing one job, but doing that job well.

utilities waiting to be judged worthy of inclusion. We especially hope **haschanged** makes it, as this creates a hash of a file when first run and then checks whether the hash has changed when run subsequently.

Project website
<https://joejh.name/code/moreutils/>

Google Drive client

GoSync 0.4

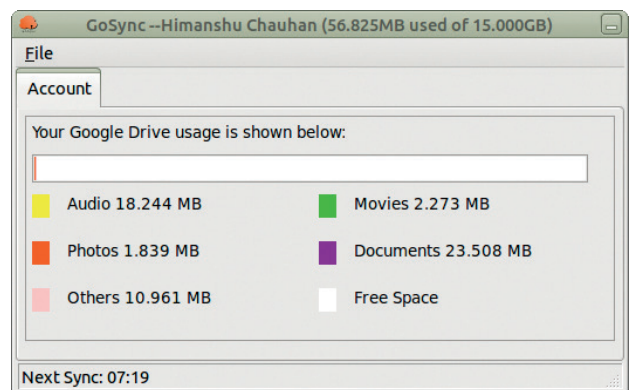
We’ve taken a step back from public cloud services like Dropbox, partly because of privacy concerns, but also because we don’t want to become reliant on something that may become expensive. OwnCloud/NextCloud has been our drop-in replacement – it’s open source, provides many of the same facilities, and can be run on a cheap VPS and even Amazon services for very little money.

We’ve even had success running the server off a modest Raspberry Pi 2 at home, although the storage access through USB does become the bottleneck. But we must admit that we also use Google’s Drive service for convenience, especially for those documents scanned with the mobile application and for large binary objects we know we’ll only

need temporary access to. This has meant that the lack of an official Linux client – long promised by Google – has affected its usability. **GoSync** is the latest third-party client we’ve spied, trying to fill this hole. It’s built using Python 2.7, and we’d recommend using Python’s **pip** tool to install it and its various dependencies.

The only thing you need to be careful of is any conflict with a Python 3 installation, as this is now the default, but most distributions have a solution for running both (they may use **pip2** as the install command, for example). You also

The lack of an official Linux client for Google Drive has affected its usability



If you’re still waiting for a GUI to Google Drive, try **GoSync** [image: Himanshu Chauhan].

need to go through the steps to create your own **clients+secrets.json** file, which means getting an API key from Google. This is straightforward, although the **GoSync** docs avoid telling you how to do this directly (although it does offer hints when you first launch it).

Project website
<https://github.com/hschauhan/gosync/releases/tag/v0.4>



Two-factor authentication

FreeOTP+

We'd argue that two-factor authentication is now essential, even if you're not obsessed with security. With so many websites spewing so many login details and so much personal information, adding a second factor to the process of logging in is a good way of giving yourself more control and security.

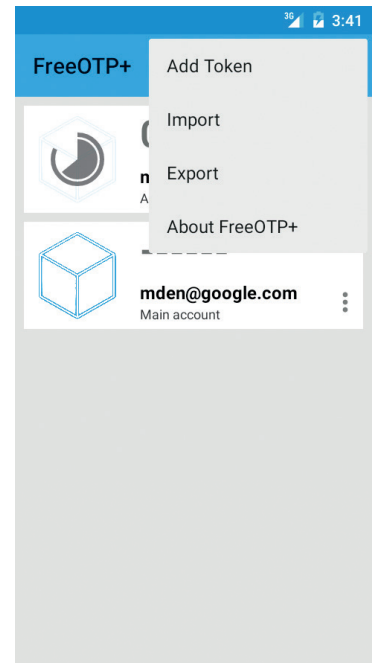
One of the most common methods is through *Google Authenticator*, a tool that generates a one-time password (OTP) that lasts only a short time. The password is generated from a key that you add to the authenticator, and while the application is now proprietary, older versions were open source and have been forked into various new tools.

Our favourite has always been *FreeOTP* (now hosted by the Fedora Project), an open source implementation for Android that

you can install through F-Droid. The only problem with *FreeOTP* is that you couldn't easily back up your keys, which meant if you lost or broke your phone, you'd become locked out of the accounts you've secured with *FreeOTP*. This is where a new fork of *FreeOTP* delivers – *FreeOTP+*. It adds what we'd argue is an essential function – backing up your key database.

By default, it can save a JSON file with these details to your Google Drive, which is probably wise, as you're already likely to be securing your Google account with two-factor authentication. You definitely don't want this file to be accessible, so encrypt it if you're storing it locally, which is also an option. Then, if the worst should happen, you won't need to beg your account holders to re-instate your access – just install *FreeOTP+* and restore your keys from the backup.

We loved the original open source *FreeOTP* authenticator, and the simple features in this new version make it work the upgrade.



Project website

<https://github.com/helloworld1/FreeOTPlus>


Open source maps

OsmAnd~

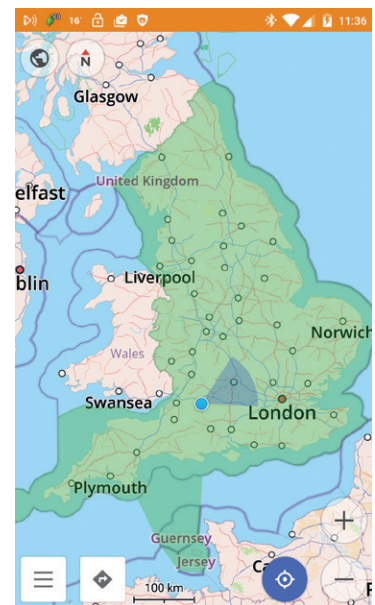
There can't have been many Android users who haven't come across this wonderful app. It's a portal to the world of open source maps and a great tool for finding your way around, whether on foot, on bike or in a car. But its best feature is that it's open source, and so are the maps, mostly pooling resources from OpenStreetMap, but also sites such as Wikipedia for other information.

Unlike most other mapping applications, all this data is downloaded for offline using, which is perfect if you're travelling in locations with extortionate data fees. We'd highly recommend paying for this app through the Google Play store, which side-steps a download limit on the demo version. But as the app is also

open source, it's available through F-Droid with no download limits or restrictions. This is why there's a tilde ~ symbol in the project's name – this character is used to differentiate an open source build from the commercial package, which adds a + to its name.

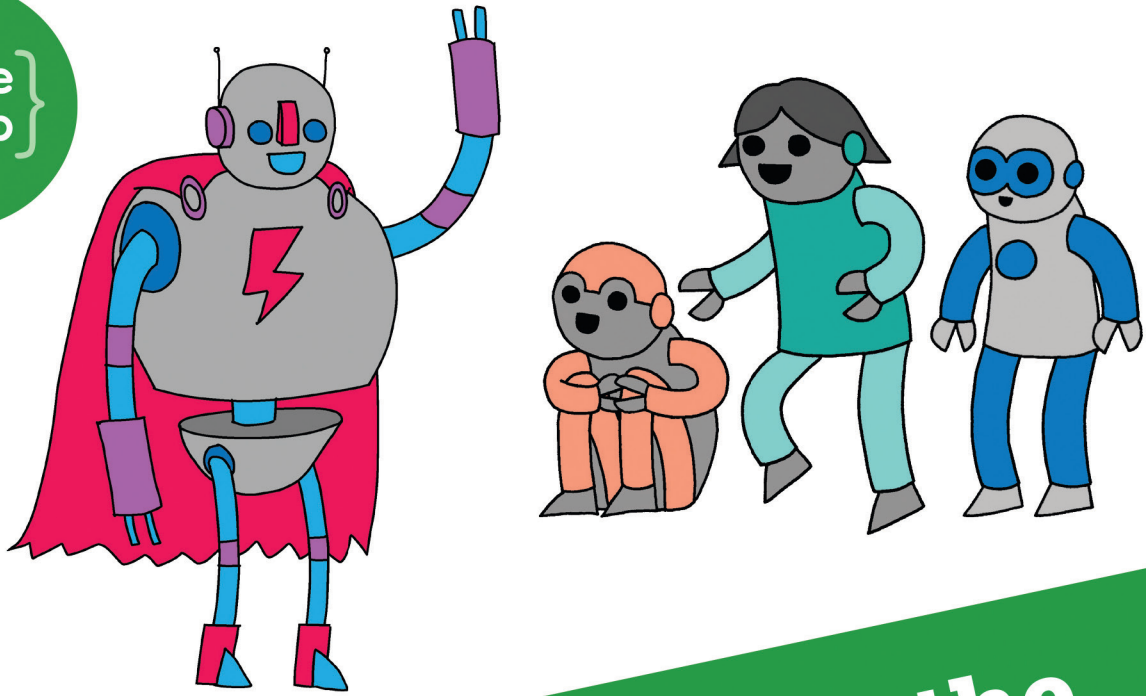
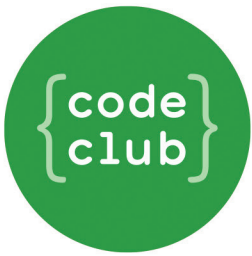
Either way, the app is full of features, from the compass view that changes according to the direction you're pointing, the plugin system with view for skiing and sailing and the ability to turn on and off the many OpenStreetMap layers. You can even just download the roads, if you're sticking with the car, and this provision is one of the reasons we'd recommend paying as the cost of the infrastructure behind the map provision must be considerable. 📍

The latest version of *OsmAnd* even includes turn-by-turn road navigation, complete with recorded or synthesized voice.

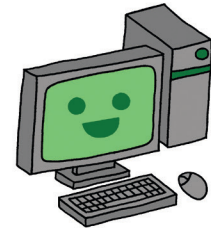


Project website

<http://osmand.net/>



Can you help inspire the next generation of coders?



Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at www.codeclub.org.uk

TUTORIALS

Warning: excessive Linux knowledge may lead to fun and more efficient computing.



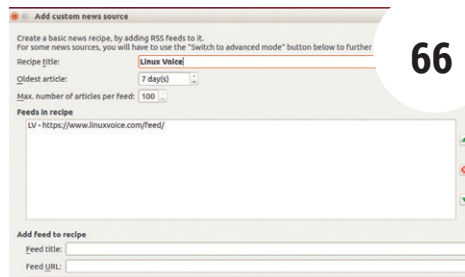
Mike Saunders
Has almost finished porting Systemd to MikeOS.

Someone asked me the other day what my first experience of programming was. It took me a while to remember exactly, but then my neurons finally aligned: it was typing out the BASIC source code for a *Breakout*-like game from the ZX Spectrum +2A manual (*Bustout*, at <http://tinyurl.com/jo2f65c>). At the time I had very little idea what the code did, and I was only seven or eight years old, but I remember throwing a massive hissy-fit when the game wouldn't work properly. Luckily, my brother went through the code I'd typed in and found the mistake. How right Linus was about many eyeballs making bugs shallow...

People often deride BASIC for being a bad starter language, and it's true that ZX Spectrum BASIC with its line numbers and GOTOs was an awful language that encouraged spaghetti code. But it really encouraged me to go further, try new languages, and poke around inside the machine. I then moved on to the Amiga (let me wipe a tear from my eye) and then Linux. Who would've thought that I'd write my own BASIC interpreter in assembly language for my own OS 25 years later...

mike@linuxvoice.com

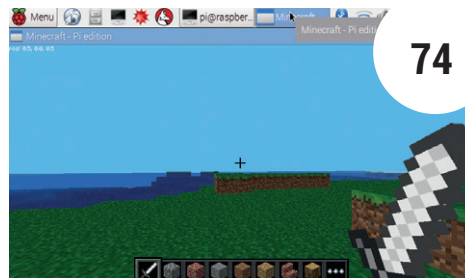
In this issue . . .



66

Create your own news site with Calibre

Ben Everard uses *Calibre* to condense the news, so he doesn't get too distracted by the rolling farce that is the United Kingdom these days.

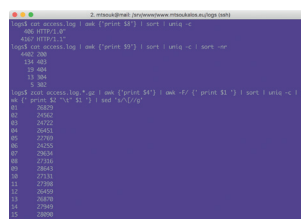


74

Raspberry Pi: Input data with barcodes

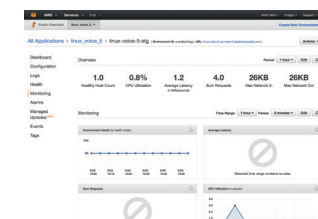
Les Pounder shows how the humble barcode can be used to do much more than tell us how much a can of beans costs.

Coding



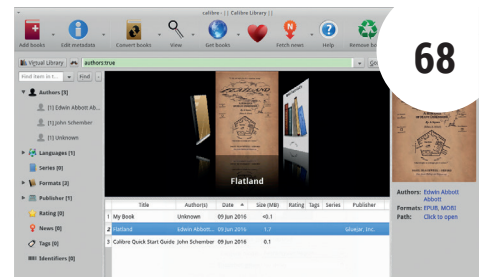
Coding: log files
Use AWK and R to reveal information from your Apache logs, with **Mihalis Tsoukalos**.

82



Amazon Beanstalk
Amit Saha hosts and scales a Golang web application on Amazon Elastic Beanstalk.

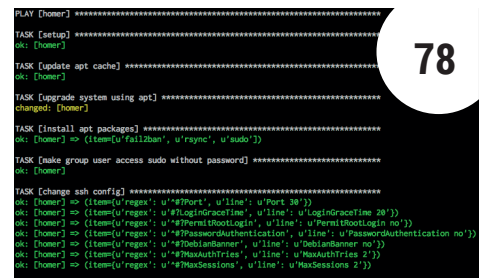
86



68

Publish with Free Software

Publish to suit your style, use FOSS tools to free your creations, and transform text – also with *Calibre*. **Andrew Conway** explains all.



78

Harden many servers with Ansible

Sebastian Götttschkes takes you through Ansible, a great way to prevent the most common attacks against web servers.

Get access to every Linux Voice tutorial ever published in our digital library of back-issues available exclusively to subscribers – turn to page p56 to join.

CREATE YOUR OWN NEWS WEBSITE WITH CALIBRE

Wrap up the latest news as an eBook and share them on your personal website.

BEN EVERARD

Why do this?

- Deprive Murdoch of advertising revenue
- Save bandwidth
- Avoid having to be confronted with opinions that differ from your own

The web has made a phenomenal amount of information available, much of it up-to-the-minute news. However, web browsers don't always make good environments for reading the news. There can be flashing adverts, pages can be slow to load and jump around as you read them, and you may want to read the news when you're offline. In this tutorial, we're going to get around all these problems by building a system that automatically

downloads the latest news from whichever news sites you want, packages these stories up as an ePub, then serves these ePubs on your own website so you can grab them from your portable devices.

A word of caution though: the more you focus on what you want to read, the less you'll accidentally stumble across things that you don't want to read, but might find interesting. Beware of trapping yourself in your own information bubble!

STEP BY STEP: PUBLISH YOUR PERSONAL NEWS

1 Install Calibre

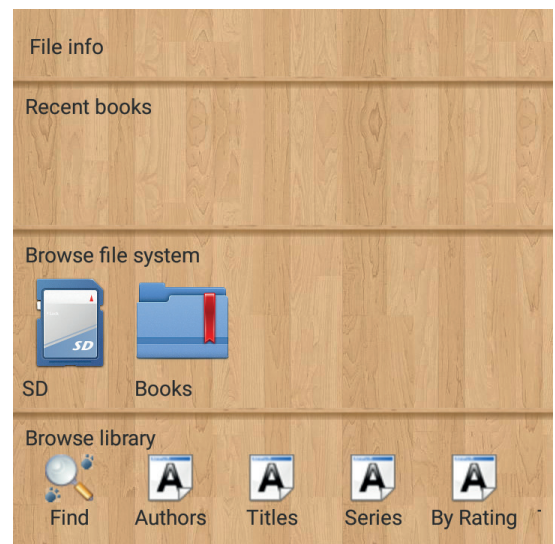
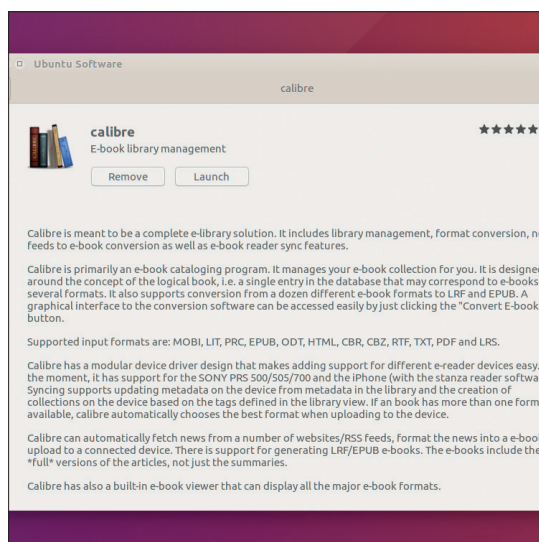
The first thing we're going to need is the software to run our newspaper website. In this case, it's just a single application that does everything: *Calibre*. If you run a server, that's ideal; if not, this can be your home PC, but you'll only be able to use the website when the PC's switched on. If you want to be able to access your website from outside of your LAN, you'll need to set up port forwarding on your router and dynamic DNS – see your router's documentation for details of how to do this.

You can use *Calibre* to manage your digital library as well as running your news website. If you're a Linux Voice subscriber, you can import our ePubs and make them available on all your devices.

2 Get an eReader

We're going to convert the news sources into ePub format, so you're going to need a way to read these. This doesn't have to be specialist hardware – there are plenty of eReaders available for Android and iOS devices as well. *Cool Reader* for Android is open source (available via F-Droid or the Google Play store), and a capable reader for phones and tablets. Adjusting the font and background of the reader may make reading easier on your eyes.

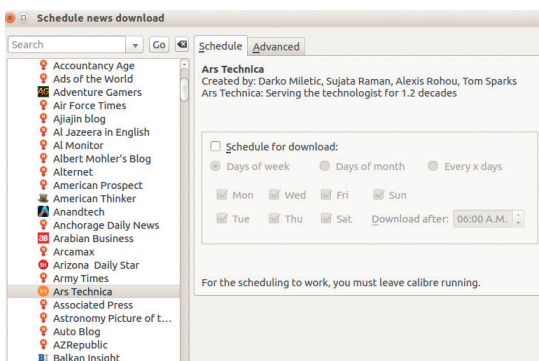
For the best reading experience, eReader hardware has eInk screens that put less strain on your eyes. Any of the eReaders that support ePubs will work with this setup provided they have a web browser to enable them to download the files.



3 Set up news collection

Now you've got all the software you need, it's time to configure *Calibre* to download the news you want. Websites aren't usually designed to be converted into ePub format, so the software needs to know how the site is structured in order to get the latest information without clogging up your computer with the entire content from the site. Fortunately, *Calibre* comes with over 1,500 news sources already configured, so there's a pretty good chance it knows how to download the items you want.

Click on the Fetch News button, and find the news source you want in the list. Check the Schedule For Download box and set the download frequency (by default, this will be every day). Once you hit Save, *Calibre* will add that site to your schedule and start grabbing the news.

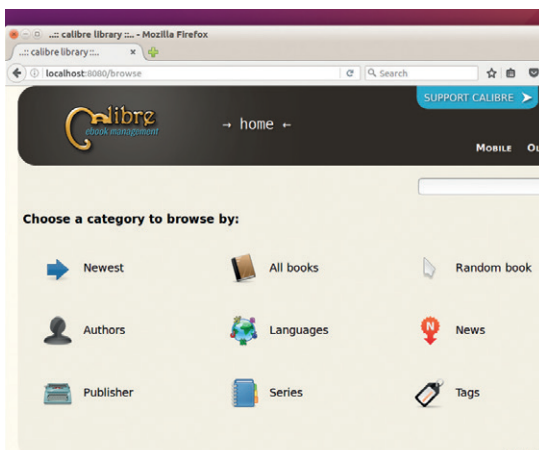


5 Read and enjoy

Your library is now online. Head to **localhost:8080** in your web browser on the same machine *Calibre* runs on to make sure everything's set up. From other machines on the same local network, you can access your *Calibre* library via the IP address of the server. Enter the command **ip addr** in a terminal and you'll get lots of details about the machine's connection. Look for the block that details your network connection, and in that, you'll see a line that starts with **inet** followed by four numbers separated by full stops. For example:

```
inet 192.168.0.19/24
```

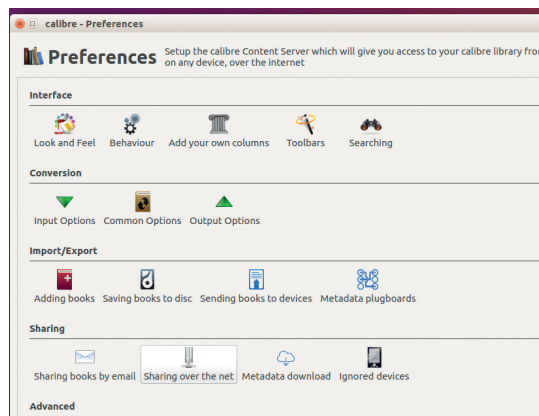
In this case, you can use the URL **http://192.168.0.19:8080** to access your library from other machines on the same local network.



4 Start the web server

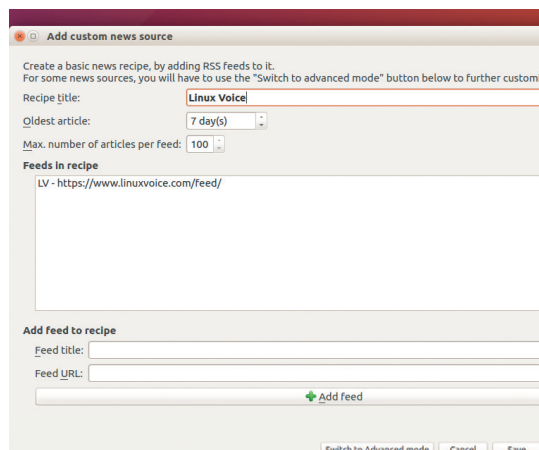
Calibre should start the first news download shortly after it's set up, so the next thing we need is the web server running. You can do this in two ways, either through the GUI or from the command line. In the GUI, go to Preferences > Sharing Over The Net to open the web server options window. The defaults should be fine unless you want to password-protect your library, or if you want to run on a different port because you already have a server on 8080. Press the Start Server button to get online.

You can also use the **calibre-server** command to launch the web server from the command line. It doesn't daemonise, so you may want to run it via **tmux** or create an *init/Systemd* script.



6 (Optional) Create a custom news source

We've looked at adding a news source that *Calibre* already knows, but you can also add any site you want. Click on the drop-down arrow next to the Fetch News button and select Add A Custom News Source. The easiest way to add a news source is via RSS. For example, to add a news source for **LinuxVoice.com**, give the recipe a title, then you just need to add a single feed. Add the feed URL **https://www.linuxvoice.com/feed/** with another title, click on Add Feed, then Save. Creating the feed won't automatically schedule a download, so you'll have to add your new source as per step 3. If the source you want doesn't have RSS, you can still add it, but you'll need to use Advanced Mode, which involves Python coding.



PUBLISH FREELY WITH CALIBRE

Influence global ideas by self-publishing your scibbles on Free Software.

ANDREW CONWAY

Why do this?

- Publish to suit your style
- Use FOSS tools and free your creations
- Transform your texts with Calibre

Calibre is licensed under the GNU GPL v3 and is available on Linux via your distro's repositories, but builds are also available for Mac OS X and Windows.

You can use one of many good word processors to do your writing, or you can work in plain text with formatting specified by markup (or down) languages and an array of free and open source software tools. With these you can bang out the words and create your very own *magnum opus*, but what good would that be if no-one ever reads it?

There are two main routes to publishing these days: you need to either find a publisher or self-publish. Either way, at some point you'll have to decide on a format for your book. Even if you're aiming for a dead-tree version of your book, these days people will expect there to be an eBook too, so it's something that cannot be avoided.

First off, if you're seeking a publisher, especially if you're unpublished, you must submit exactly what a publisher asks for, which is usually specified on their

website somewhere. So have a look, and if they ask for a stack of papyrus with hieroglyphs from a particular Egyptian dynastic period, then that is what you should send them. Thankfully, most publishers aren't that fussy (Linux Voice accepts submissions from any dynastic period) but most first-time authors are still asked to provide a chapter or two of their text printed on paper. The idea is that emailing an electronic copy is too easy and publishers like to erect some barriers to deter half-hearted submissions.

But, for whatever reason, it is likely that you'll have your text in one format and find that you need to provide it in some other format. Last issue we went through the process of creating text in markdown and then turning it into HTML and CSS, and we showed how this could be manually turned into an EPUB file, one of the most popular eBook formats.

In this article we'll look at two related things: how to

	Title	Author(s)	Date	Size (MB)	Rating	Tags	Series	Publisher
1	My Book	Unknown	09 Jun 2016	<0.1				
2	Flatland	Edwin Abbott...	09 Jun 2016	1.7				Gluejar, Inc.
3	Calibre Quick Start Guide	John Schember	09 Jun 2016	0.1				

convert formats, and also some platforms on which you can self-publish. We'll also pay some attention to formatting for print because, as mentioned above, this is still often required when seeking a publisher.

Where to self-publish

We were genuinely amazed at how many ePublishing platforms there are these days. It seems that many an investor has parted with money to fund entrepreneurs and their start-ups that cater for every conceivable ePublication niche. We can't cover them all, so we've picked three that represent three quite distinct parts of the spectrum. Notice that we use the word 'platform' rather than 'publisher' at times – the distinction may be a little pedantic, but part of the meaning of the word publisher is to publicise your work. With self-publishing, the onus is really on the author to push their work out into the world, making best use of the many self-publishing platforms out there, and, as most are not exclusive, there's nothing to stop you selling on many simultaneously.

Amazon KDP

Let's start with the most famous: Amazon Kindle Direct Publishing (KDP). The most obvious reason to go with this big corporate is that it is so well known. The second reason is that it offers a higher percentage of royalties than a traditional publisher but, as we'll see, not the highest among ePublishing platforms. (Royalties are the percentage of money from sales that go to the book's author.) A book I've just published on Amazon gets 70% royalties, whereas one that I wrote over 10 years ago, prior to the ePublishing revolution, gets just 10% royalties. But don't forget, with self-publishing you have to put in the work that a traditional publisher might have done for you, otherwise you may end up with 70% of nothing.

The technical process of publishing on Amazon is not too difficult, though it is difficult to do well. There are subtle aesthetics that can impact both sales, like a poor cover, and the reading experience, such as layout and appearance, and some technical details such as

the table of contents can be tricky to get right. To be fair, this is true of any ePublishing platform, but the downside of KDP making it very easy to publish is that it lulls authors into a false sense that all aspects are easy, and there are quite a few books published using it where the author has not taken enough care to get the details right. That said, Amazon has made an effort to help authors improve quality, such as an automatic spell-check of the entire text when you upload your book. Our main advice for KDP, and in fact for all platforms out there, is to check that the finished

The technical process of publishing on Amazon is not too difficult, though it is difficult to do well

product looks acceptable on as many eReading devices as you can, including phones and tablets.

To get started with KDP, go to kdp.amazon.com and have a read of the documentation there. There's a 100 page *Amazon Kindle Publishing Guidelines* document that's worth at least a skim. Unfortunately, although there is a lot of very good information in that guide, and elsewhere, not all of it was up to date (advice on the use of UTF-8 character sets was conflicting, for example). Nevertheless, if you've read the basics, it is quite feasible to go from signing up to KDP to having your book live and selling within 24 hours (not including writing the book of course!).

One of the best ways to prepare a book for uploading to KDP is to use Amazon's *Kindlegen* tool. Happily, this tool is available for Linux via a binary tarball download, although the software itself is not released under a FOSS licence. We downloaded it and got it working without any dependency issues. If your book is just in one file called **book.html** in directory **/home/fred/books**, then a command as simple as this can do the trick from the directory where the **kindlegen** binary is located:

```
./kindlegen /home/fred/books/book.html
```

Comparison of eBook platforms

Website	Input formats	Download formats	Monies to author	DRM	Exclusive
kdp.amazon.com	HTML,EPUB,MOB	Via kindle app	70% royalties	Up to author	Up to author
leanpub.com	markdown,DOCX	EPUB,PDF,MOBI	90% minus \$0.5 per transaction	No	No
unglue.it	*no conversion*	EPUB,PDF,MOBI	92% minus \$0.25 per transaction	No	No



Calibre quick tour

Calibre's interface is divided into four main areas. The main one in the middle shows you books in your library, and you can use the left-hand panel to filter them by author, tag, language and more. The panel on the right gives you some brief information about the book including a list of available formats.

For *Flatland* we have the downloaded EPUB and the MOBI format that Calibre created for us. The button panel at the top enables you to perform common tasks such as adding books and converting them between formats. Most of those functions are also available via a right-click on a book in the main

panel, which brings up a context menu. At the bottom-right you can see the Jobs indicator, which displays progress of time-consuming tasks. To the left of that are some icons that control the layout of Calibre's display, which toggle the visibilities of the three panes below the buttons.

The screenshot shows the Calibre Library application window. At the top is a toolbar with icons for 'Add books', 'Edit metadata', 'Convert books', 'View', 'Get books', 'Fetch news', 'Help', 'Remove books', and 'Calibre Library'. Below the toolbar is a search bar with 'authors:true' entered. A left sidebar contains a 'Find item in...' dropdown and a 'Find' button, followed by filter categories: Authors [3], Languages [1], Series [0], Formats [3], Publisher [1], Rating [0], News [0], Tags [0], and Identifiers [0]. The main area displays a list of books with columns for Title, Author(s), Date, Size (MB), Rating, Tags, Series, and Publisher. The book 'Flatland' is selected, and a right sidebar shows details for 'Flatland', including the author 'Edwin Abbott Abbott', formats 'EPUB, MOBI', and a path to open the book.

	Title	Author(s)	Date	Size (MB)	Rating	Tags	Series	Publisher
1	My Book	Unknown	09 Jun 2016	<0.1				
2	Flatland	Edwin Abbott...	09 Jun 2016	1.7				Gluejar, Inc.
3	Calibre Quick Start Guide	John Schember	09 Jun 2016	0.1				

This will output the file **book.mobi**, which you upload to the KDP website and publish. *Kindlegen* can convert books of much greater complexity from an EPUB or from a directory or Zip of HTML and image files. If you followed last issue's tutorial then you can either give the EPUB file directly to the **kindlegen** command, or skip the EPUB creation and just give it the **content.opf** file, from which it will find all HTML content files, image files, the CSS file (for styling) and, if present, **.ncx** or other files for generating the table of contents.

If you've written your book using a word processor, such as *LibreOffice Writer* or *Microsoft Word*, then you'll need to export it as an HTML file first. We've not tangled with this method, but there is some advice on how to do it on the KDP website.

To our surprise, KDP places few restrictions on exclusivity. You can publish your book elsewhere and at different prices, but Amazon warns you that they may drop the price to match a lower one offered elsewhere, and they also try to tempt you to be exclusive with enhanced promotional benefits. We were also pleased to discover that the author gets to decide whether Digital Rights Management (DRM) is applied to their book. There's no obvious reason why KDP's terms and conditions would prevent use of Creative Commons licences or the GNU Free

Document Licence, but neither have Amazon given clear answers on this point when asked. What is crucial, and indeed important on any platform, is that you own the copyright for the text.

Leanpub

Leanpub's strapline is "Publish early, publish often". This may give you a clue as to the intended audience: folk who like developing in the open. Unlike Amazon, Leanpub is not just about publishing to a mass market and making the author many sponduliks; rather it provides a platform for the author to write and develop her or his book and, if they choose, to do so within a community. To this end Leanpub's platform is designed to make it easy for readers to contact the author and make comments on books on dedicated feedback pages. They also encourage authors to put works in progress on Leanpub; in fact, that's part of its *raison d'être*.

Unsurprisingly, Leanpub's catalogue contains many books on technical subjects related to writing code, especially in a FOSS context. Leanpub's very first book, *Startup Lessons Learned* was written by one of the proponents of the Lean startup model for fledgling companies, from which Leanpub got its name.

Leanpub is competitive with their royalties, offering authors 90% less 50 cents per transaction. Not only

3 ways we can make ebooks free



Unglu.it offers three payment models, so you can choose the one that fits best with your commercial goals.

that, but you can set a recommended and minimum price and then let potential readers pay whatever they wish. The minimum price can even be zero with the 50 cents fee waived. We've used Leanpub and while we saw some readers nab a book for free (fine, because we said they could!), we also saw others pay over the recommended price. As Humble Bundle demonstrated for games, and to a lesser extent for eBooks, allowing users to set their own prices does not necessarily lead to a crash in prices.

Plain text using Markdown is the preferred writing format, and they've created their own flavour of Markdown called Markua. There's even a book on the Markua specification published on Leanpub. You can either write using their online editor or else on your computer using any text editor you like and use their integration with Dropbox or GitHub to synchronise your local copy with the one on their website. Either way, when you make a change to your book you can, at the press of a button, create PDF, EPUB and MOBI versions and peruse an online preview. Overall, if you're comfortable with Markdown, this is even easier than using Amazon KDP's tools. Also, if Markdown is not to your taste and you prefer working with *Writer* or *Word*, then you can opt to save a .docx file to a Dropbox folder and have that transformed into the above formats.

As you might expect, Leanpub doesn't stipulate any licensing and doesn't mind if you make your book available elsewhere.

Unglue.it

Unglue.it aims to free books by getting them released under one of the Creative Commons licences – to free them from being stuck in the "glue" of traditional publishing and copyright restrictions. It serves the three functions you might expect of an ePublishing platform: it helps you to publicise your book, it enables people to download it, and you can make money. The copyright remains with the author and you can't unglue a book without the copyright holder's consent.

There are three routes to ungluing your eBook. The first is pledge-to-unglue, in which individuals each pledge an amount of money and once a target set by the author is reached, the pledgers' credit cards are

charged and anyone can read the book for free, and remix it within the terms of the chosen licence. The second method, buy-to-unglue, is similar, except that instead of pledging, individuals buy copies of the eBook and each purchase moves the book closer to being unglued once the target is reached. Both of these methods are essentially variants on crowdfunding except that the book has to be complete at the outset.

The third method, thanks-for-ungluing, is different. Here the author releases the book under a Creative Commons licence first and then simply asks supporters to make donations as a gesture of appreciation. Needless to say, whatever method is chosen, the end result is that the book becomes freely available in electronic format without any DRM. EPUB, PDF and MOBI formats are supported. Clear instructions for downloading books to your eReader are given, and if you sign up for an unglue.it account you can send books direct to your Kindle.

For each contribution that is made to a book (there's a minimum of \$1), Unglue.it takes \$0.25 plus 8%, and the remainder goes to the creator of the work. However, Unglue.it is part of a not-for-profit corporation called the Free Ebook Foundation, so the proceeds it collects will further the cause of freeing eBooks. Overall, this is an excellent way to promote

Unglue.it aims to free books by getting them released under one of the Creative Commons licences

freedom in creative works and in addition to any new books that are released freely, it offers a new electronic lease of life to a sea of out-of-print books that publishers have lost interest in.

Many others

There are of course many more platforms for launching your ebook, including offerings from big players such as Apple with iBooks, Google Play, and smaller but significant platforms such as Smashwords. In addition to eBooks there are some

that specialise in self-publishing to print, such as Amazon's CreateSpace or Lulu.

Many of the services out there, including two of the ones mentioned above, provide ways of converting to common eBook formats. However, if you want more control of details – and as mentioned above, details such as the cover and styling are important – then you'll want to perform the conversion process yourself. If you're not averse to a bit of XML you can craft an EPUB manually, but XML is, almost by design, a human-unfriendly markup language, so most of us will be more comfortable using an application that will automate it, and the FOSS star on this stage is *Calibre*.

Calibre is an application that manages libraries of eBooks and enables you to convert between a wide array of formats. There are also many plugins that add to the basic functionality: these range from statistics functions on wordcounts to plugins that can remove DRM from eBooks.

The latest versions of *Calibre* are built around the *Qt 5* framework and look slick as a result, though its button-cum-menu interface might feel a little odd at first sight. See the boxout for a quick tour of its main window.

Calibre comes pre-installed with one book: its own Quick Start Guide. Double-click on it in the middle

Most of us will be more comfortable using an application that will automate the creation of our EPUB file

pane and it will open in *Calibre's* eBook viewer. This is worth a few moments of your time, and at only 37 pages it doesn't take long to flick through.

Adding books to *Calibre* is easy enough. Click on the Add Books button at the top-left and a file browser window will open. Select any book file that you happen to have lying around on your computer and *Calibre* will pull it into its library. This copies book files into the Calibre Library directory, which is placed in your home directory. If you don't have any book files, then we recommend downloading the book *Flatland* from unglue.it – a book written about what it'd be like to live in a two-dimensional world, written by Edwin Abbott in 1884.

Converting eBooks is straightforward. We downloaded the EPUB of *Flatland* for this example, but the same principle applies to any book and in any format. Select the book in the main pane by clicking once on it, and then click on the Convert Books (or you can select it via a right-click context menu). The conversion window opens and offers you a bewildering array of things you can change, from the cover image and metadata, all the way down to making search and replacements in the text with regular expressions. For now, don't worry about any of that, just notice that at the top-left it says the Input Format is EPUB and at the top-right the Output

Format is MOBI. To do the conversion, click on the OK button at the bottom-right and after a few moments of the 'Jobs' indicator at the bottom-right whirring round, you'll have a MOBI edition of the book. You'll see that EPUB and MOBI are both listed as formats in the right-hand pane.


You can save the EPUB or MOBI or any other format that's available via the right-click context menu, but you can also just go to the Calibre Library directory in your file manager (or on the command line) and find the **.epub** or **.mobi** files there.

If you've constructed a book from source in HTML then you can use *Calibre* to assemble it into a full eBook in a format of your choosing. If you don't have such a book to hand, download the very simple **mybook** example (it's only got one page) from the previous article from here: github.com/mcnalu/linuxvoice-publishing. Click the Add button and select the HTML file you wish to use, and *Calibre* will pull it into its library.

At this point the book will only be held in ZIP format, and if you try and view it, it'll just show you the files within it. If you look closely, you'll notice that *Calibre* had spotted that a CSS file is specified and brought that into the library too. To turn the book into a proper eBook format, just hit the Convert button as described above. Once this is done you'll see the new format (MOBI by default) listed on the right, and viewing the book will now bring it up in *Calibre's* viewer.

Write on, commander!

With software such as *Calibre*, you can take charge of your writing. You can alter the look and feel, edit the CSS and even take advantage of its 'heuristic processing' feature to automate mundane tasks that may take hours to do by hand, such as removing blank lines and dealing with unwanted wrapping of lines. It can also detect the structure of your book and generate a table of contents for it. This is all possible with command-line-fu, but the beauty of *Calibre* is it gives some of this power to non-technical users.

Despite all the applications and services that allow you to create and publish your book to the world, there is still one bit that no computer can yet help you with: the creative process of writing. For most authors – at least ones who aren't simply doing it for money – they write because there's something they feel they must express. Learning how to perform some technical task, such as creating an eBook, is interesting in itself, but for many of us it is not quite enough; we must also share the knowledge, be it in book, blog or forum post. It is said that everyone has a book in them. Whether that's true or not, it's certainly worth asking yourself if you have some knowledge you'd like to share. The art of good writing may remain as challenging as it every was, but the process of publishing a book has never been easier. 

Andrew Conway watches the stars from his wood-panelled study. He likes open data and what you can do with it using Free Software.

Emergency

→ **Ebola**

↑ **Gunshots**

→ **Landmine**

↗ **Cholera**

↖ **Shrapnel**

← **Maternity**



The world's A&E Department

Find out more at msf.org.uk

BARCODE INTERFACES WITH EASYGUI

The humble barcode can do much more than tell us how much a tin of beans costs.

LES POUNDER

Why do this?

- Add a new input method to your projects
- Use lasers!

You will need

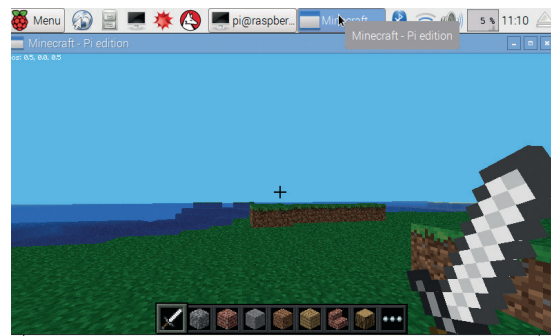
- Any model Raspberry Pi running the latest Raspbian release
- A barcode scanner
- Female–male jumper wire
- 3 x LED (red, amber, green)
- A breadboard
- 3 x 220Ω resistor (red, red, brown)

The circuit for our flashing LED code is relatively simple, requiring only a few cost-effective components to build.

The humble barcode is all around us. Our parcels, beans and medicines are all catalogued using a series of thick and thin lines. In issue 20 we used barcodes from store cupboard staples to create an adventure game similar to Pokemon, you can find the code for this at https://github.com/lesp/LinuxVoice_Issue_20.

Barcodes are great fun, and they can also be used as a simple form of input. Using a barcode scanner purchased from Amazon for around £30 we can create a novel method of input that any child can operate. The scanner works as a true plug-and-play device, requiring no installation or drivers; it simply appears as a human interface device similar to a keyboard. Once a barcode is scanned the value is decoded and sent as standard input to the computer; this is followed by a virtual Enter keypress that will enter the value.

There are many types of barcodes, including the EAN13, CODE39 and ISBN variants. EAN13 is used in retail, so you will see this barcode on your tins of beans. CODE39 is used by parcel companies, and ISBN is used by publishers to catalogue books, and is commonly used on Amazon to search its vast database of books. Creating your own barcodes is



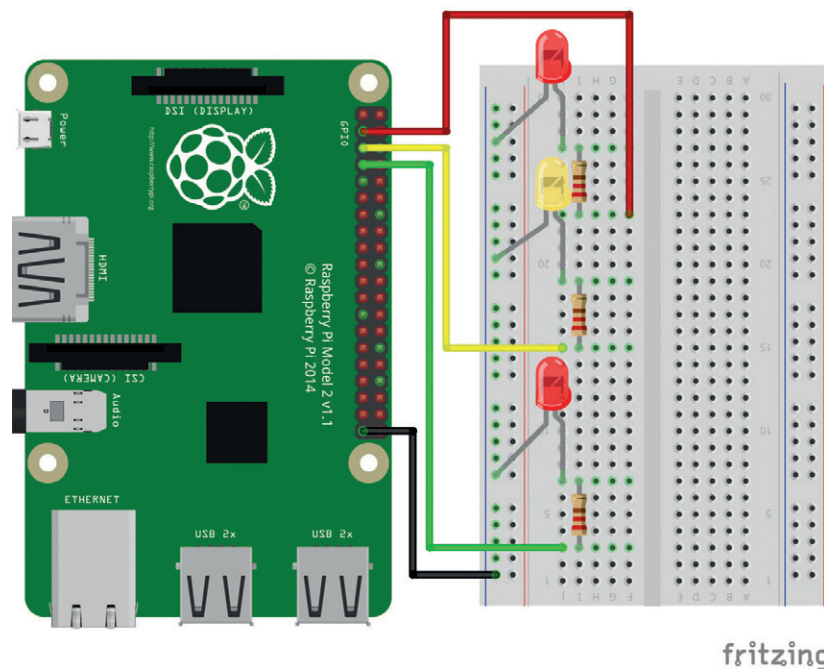
Playing *Minecraft* by scanning a barcode is just one of the many actions that we can trigger with our new toy.

really easy thanks to online services such as <http://www.barcode-generator.org>. We chose to create CODE39 barcodes and then save them as SVG files. CODE39 barcodes are incredibly flexible and permit the use of alphanumeric characters, enabling us to add text to our barcodes. With a little help from *Inkscape*, <https://inkscape.org/en> we created a series of A5-sized sheets that contained all of the barcodes for a particular game. We have included our cards as part of the downloads for this project.

The barcodes created can be used to control or trigger many different aspects of a project. For example, we could have a barcode that triggers a robot to drive around: the reader can be mounted on top of the robot and it can scan the floor before it. Once it detects a barcode it can react by turning in another direction, playing a sound or flashing some lights. Let's get started and build a new game using our own custom barcodes and a little Python code.

Hardware

Our hardware installation is rather simple and comprises two parts. First, we can plug in the USB barcode scanner into a spare USB port on our Raspberry Pi. Second, we'll need to build the circuit for our flashing LEDs. This circuit is relatively complex, requiring that we connect each of the LEDs to a GPIO (General Purpose Input Output) pin. We attached the long leg of each LED, red to pin 2, yellow to pin 3 and green to pin 4 via a resistor to limit the current and using male–female jumper wires. Please see the diagram for a suggested layout – a high-resolution



fritzing

version is available via the software download for this project. Once the wiring is ready, attach all the accessories required to use your Raspberry Pi and boot to the Raspbian desktop.

Software

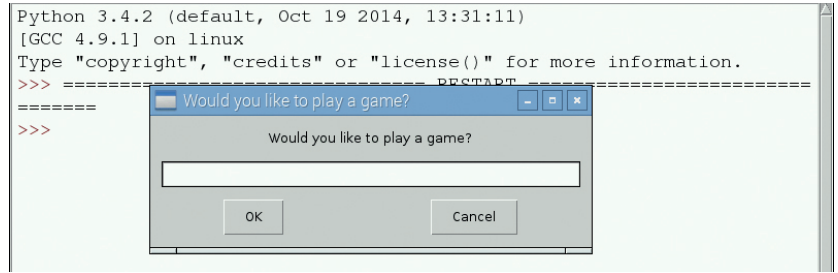
We start the software section of this project by opening the Python 3 editor, which can be found in the main menu at the top-left of the screen. In the menu look for Programming and then Python 3. Once the editor is open, click on File > New and create a new blank document. It is best practice to save often, so we shall save the blank document to ensure that subsequent saves are quicker. Click on File > Save and save your project as **Barcode-project.py**.

This project is coded entirely in Python 3 and uses a number of modules, sometimes referred to as "libraries". The majority of the modules that we use are installed by default on Raspbian, the only exception being the EasyGUI module, which we will need to install manually. We shall open a terminal – you can find the icon for this on the Raspbian desktop – at the top-left of the screen looking like a small computer monitor. Open the terminal and run the following command, press Enter to execute.

```
$ sudo pip3 install easygui
```

Once this command has completed, you can close the terminal and return to the Python editor.

To work with modules we need to import them, and we do that first of all. Our first module is called **subprocess**, which is used to interact with the underlying Raspbian operating system; we shall use it to call commands via the Terminal. The next module is called **Time**, which is used to control the pace of our project, for example setting delays for how long an LED should be on/off for. We import the EasyGUI module but rename it to **eg**; this shortens the rather long module name. Our next import is GPIO Zero, and from it we import the class **TrafficLights**. This class can be used with three LEDs and control them in a similar manner to common traffic lights. Our next two imports are **Pygame** and **Pygame.mixer**. **Pygame** is a game/media creation framework for Python, and we are using its mixer to play audio clips on demand. Our final import is the **randint**, short for random integer,



function from the Random module. We shall use it to generate a random number in our code.

```
import subprocess
import time
import easygui as eg
from gpiozero import TrafficLights
import pygame
import pygame.mixer
from random import randint
```

Our next section of code contains four functions, the first of which is an audio player. This function takes one argument, the audio file to play. Inside the function we initialise the audio mixer ready to start playback. We then load the audio file, queuing it ready for playback. Lastly we then play the audio file, with the number of times it is played being controlled by the value in brackets at the end of the line.

```
def audio(file):
    pygame.mixer.init()
    pygame.mixer.music.load(file)
    pygame.mixer.music.play(1)
```

Our second function is a number game, where a random number between one and five is chosen using the **randint** function. The random number is stored in a variable called **number**, and on the next line we create another variable called **guess** into which we store a placeholder value. We now use a while loop, this loop will repeat while a value is True. In this case it will keep repeating if the value of our guess is not equal to the randomly chosen number. Inside of this **while** loop we update the value stored in the variable **guess**; it is now used to store the answer to a question. To ask a question we use the **easygui** module, specifically the **enterbox** function. This function produces a dialog that will wait for user

EasyGUI produces dialog boxes that match the look and feel of the operating system on which it is being used.

EasyGUI

In this tutorial we used the fantastic EasyGUI to create a user interface. EasyGUI is a cross-platform module, working with Windows, Mac and Linux. We've used this module in previous tutorials as it provides an easy introduction to the various dialogs and prompts that a computer uses to communicate with the user. Using EasyGUI is relatively straightforward, and it comes with a series of dialog boxes. To see them all, open the Python 3 editor and in the Python shell type.

```
import easygui as eg
eg.egdemo()
```

The most basic dialog is a message box, or **msgbox**. This is used to advise the user. To create a basic message box type

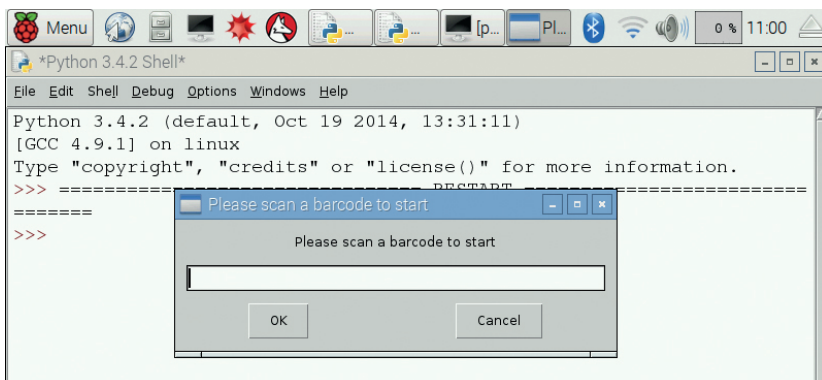
the following into the Python shell that we just opened.

```
eg.msgbox(title="Hello Reader",msg="Thanks for reading Linux Voice")
```

A little more advanced, but still easy to use, the File Open dialog box is something that we see everyday. Using EasyGUI we can create this dialog, which will capture the full path to the file that you wish to open. This can then be saved to a variable. Here we use the dialog to capture the filename that we wish to process

```
filename = eg.fileopenbox(title="Open File",msg="Please open a file")
```

You can read more about EasyGUI on its documentation page at <http://pythonhosted.org/easygui>.



When the barcode is scanned the information is automatically entered into the dialog box. It even presses the Enter key for us.

input, and this is where our barcode scanner is used. The `enterbox` function takes a number of arguments; for our project we use `title` and `msg`. `Title` creates a title for the dialog, and `msg` is used to talk to the user. In the `msg` argument we use `\n`; this is a Python instruction to create a new line, helping us to keep the

This project can be extended into an interactive media player that uses barcodes to play videos

dialog box tidy. At the very end of the `msg` we see `...when doubled it is"+str((number*2))`

This is a method of connecting the number chosen, multiplying it by two and then converting the data type to a string, as only identical data types can be joined together in this manner. By doing this we can give the user a hint to solve the math problem and keep the code agile so that it requires no manual updates. On the next line we convert the user's guess into an integer; this ensures the user only provides numerical values. As the `while` loop is still running, if the guess and the random number do not match, we call the `audio` function and instruct it to play a sound indicating a wrong answer. In the last part of the `while` loop, if the guess matches the random number then the first condition is `False`, and means that the `Else` condition is active, triggering the playback of the correct audio file.

```
def numbergame():
    number = randint(1,5)
    guess = 0
    while guess != number:
        guess = eg.enterbox(title="I'm thinking of a number
between 1 and 10",msg="What number am I thinking of? \n
Here is a hint, when doubled it is"+str((number*2)))
        guess = int(guess)
        audio("/home/pi/wrong.wav")
    else:
        audio("/home/pi/correct.wav")
```

Our third function is used to flash the LEDs attached to the GPIO. The sequence is a typical UK traffic light. To control the LEDs we use GPIO Zero and its built in `TrafficLights` class. We start the function by defining its name and then instructing the

`TrafficLights` class that we have LEDs on GPIO pins 2, 3 and 4. Pin 2 is red, 3 is yellow and 4 is green. The `TrafficLights` class expects the LEDs to be presented in that order. We then turn the green LED on ready for the sequence to start.

```
def flashing_LED():
    lights = TrafficLights(2, 3, 4)
    lights.green.on()
```

Still inside of the function we now use a `for` loop that will loop twice before exiting. Inside the `for` loop we create a sequence that will turn each of the LEDs on and off according to the sequence in which a traffic light controls traffic. To ensure that the LEDs are on and off for the correct amount of time we use `time.sleep` and pass the function the correct number of seconds to wait.

```
for i in range(2):
    time.sleep(10)
    lights.green.off()
    lights.amber.on()
    time.sleep(1)
    lights.amber.off()
    lights.red.on()
    time.sleep(10)
    lights.amber.on()
    time.sleep(1)
    lights.green.on()
    lights.amber.off()
    lights.red.off()
```

Our final function is called `quiz` and as the name suggests it is a simple quiz that asks one question.

We start the function by creating a variable called `Q` and in there we store the answer to a question asked



We imported our barcodes into `Inkscape` and made our own project cards.

GPIO Zero

GPIO Zero was introduced in late 2015 as an alternative resource to work with the Raspberry Pi GPIO. Typically a user wishing to work with the GPIO would need to use the RPi.GPIO module, which is still being actively developed and used in projects. But this module is a little tricky for those new to coding as it requires the user to know a little electronics and understand how the GPIO pins work. This can be complicated, and this is where GPIO Zero comes in.

GPIO Zero provides a number of classes and functions that enable the user to just go ahead and build a project. Using the LED class all we need to do is tell GPIO Zero which pin we have connected the LED to, then we can use functions to turn the LED on, off, fade or flash the LED with no need to use a loop. Other classes include working with motors, including a class to build a robot controlled by a user with a keyboard (or for a more advanced project sensors can be used). Sensors includes a passive infrared (PIR) which detects movement based on body heat, ultrasonics, which use ultrasonic pulses to judge distances and detect objects, and a light sensor, which can be used to measure a light level and react accordingly.

by the EasyGUI **Enterbox** function. If the answer given is not the same as the hard-coded answer, in this case **A**, then we call the audio function that we created earlier and play the wrong answer audio file. If the answer matches then the player hears the correct answer audio file.

We now move on to the main body of code and the logic that controls our project.

We start the main body with an infinite loop, **while True**. This loop will constantly run until the project is exited. Inside the loop we create a new variable called **play**, which is used to store the answer to a question, again asked via the EasyGUI enterbox function.

while True:

```
play = eg.enterbox(title="Would you like to play a game?", msg="Would you like to play a game?")
```

Now we create a new **while** loop, which will continue while the value of the variable **play** is **YES**. Inside this new loop we firstly play the 'correct answer' jingle, to indicate that the program is working. We next create a variable called **code** that will store the barcode scanned by the user.

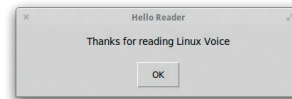
```
audio("/home/pi/correct.wav")
```

```
code = eg.enterbox(title="Please scan a barcode to start",msg="Please scan a barcode to start")
```

Still inside of the **while play == "Yes"** loop we now create a series of conditional statements. These are tests that will check the contents of the code variable against a series of hard-coded values, the same values that are encoded into the barcodes.

Our first condition to test is whether the contents of the **code** variable are the same as the value **MINECRAFT**. If that is true then the code inside of that condition is executed. So for this condition we print **MINECRAFT** to the Python shell, as a means of debugging the code. We next use the **subprocess** module and call the **minecraft-pi** command as if we were using the terminal. Lastly we break out of the

```
>>> import easygui as eg
>>> eg.msgbox(title="Hello Reader",msg="Thanks for reading Linux Voice")
```



while loop and return to the main **while True** loop.

```
if code == "MINECRAFT":
    print(code)
    subprocess.call(["minecraft-pi"])
    break
```

The next condition to test is handled using an **else if** statement, which in Python is shortened to **elif**. For the first of the **elif** conditions we compare the value of the **code** variable against the hard-coded value **SONIC PI**. If this condition is true then the code will launch the Sonic Pi application.

```
elif code == "SONIC PI":
    print(code)
    subprocess.call(["sonic-pi"])
    break
```

For the next three **elif** conditions we compare the scanned value against the hard-coded values for **LED FLASH**, **NUMBER GAME** and **QUIZ**.

```
elif code == "LED FLASH":
    print(code)
    flashing_LED()
    break
elif code == "NUMBER GAME":
    numbergame()
    break
elif code == "QUIZ":
    quiz()
    break
```

We now break out of the **if...elif** conditional statements and return to the main **while True** loop. At the start of this loop we said that while the value of the variable **play** is **YES**, run the indented code. But if the player answers no to the question "Would you like to play a game?", this condition is **False** and so the **else** condition is activated, causing the code to play the 'wrong answer' audio clip and then exit the game.

With the code completed ensure that it has been saved and that your cards are ready. Click on Run > Run Module to start the code. The first question asked will be if you would like to play the game, using the barcode scanner and the Quiz card answer **yes** to continue. The next dialog will ask you to scan a barcode to choose a game. Pick any card and scan the barcode to interact with the game.

This project can be extended into an interactive media player that uses barcodes to play videos. Imagine encoding the barcode of a book into the code and using it to trigger scenes from a play or movie based on the book! 📺

EasyGUI has many different dialog boxes and includes many advanced features that can be dropped into your project.

PRO TIP

All of the code for this project can be found at <https://github.com/lesp/LV30-BarcodeProject/archive/master.zip>

Les Pounder makes things, breaks things, and spends the rest of his time teaching teachers about the new IT curriculum.

HARDEN YOUR SERVER USING ANSIBLE

Create playbooks to be run on any number of servers to secure them.

SEBASTIAN
GÖTTSCHKES

Why do this?

- Prevent the most common attacks against web servers
- Run tasks on every server without even logging in manually
- Repeat the same tasks on new servers with a single line.

Ansible runs on your local machine and uses SSH to execute commands on remote hosts.

If you ever set up a web server and paid attention to the *iptables* logs, you might have noticed that the server was getting packets on various ports minutes after it got online. This is especially true if your IP address belongs to one of the cloud providers such as AWS, DigitalOcean or a large data centre.

It's very likely that some of those packets were sent from software looking for new servers on the internet. On each server the software finds, it tries various attacks that are known to work on unpatched software versions or for unsecure settings.

This means trouble for insecure servers, but it also means that with an up-to-date system and some basic security settings enabled, these attacks won't be successful. There are already a lot of tutorials on how to harden a server against the most common attacks. The caveat is that while most of them are not very long, the steps still need to be executed on every server you run. If there is an updated version of a specific software (like SSH or the firewall), you might need to update all those servers and maybe adjust the configuration as well. This can add up and ultimately lead to you not doing it at all. Keeping track of which configuration is in place on which server and what was installed is also a problem if the number of servers grows.

This is where *Ansible* comes in. *Ansible* describes itself as a tool for "IT-Automation", which means that after defining tasks (like running **apt-get update** on a Debian system), they can be executed on any remote server you have access to through SSH. The idea is that instead of running commands on your servers manually, they are all defined for *Ansible* and can be run multiple times, against multiple servers, to produce the same outcome. *Ansible* is run from your local machine and is agentless, meaning that there is nothing to do on the remote host in order to use it.

To get started, you need to install *Ansible* on your local machine. This can be done on Debian-based systems with

```
apt-get install ansible
```

or on any other system through **pip** (*Ansible* is written in Python):

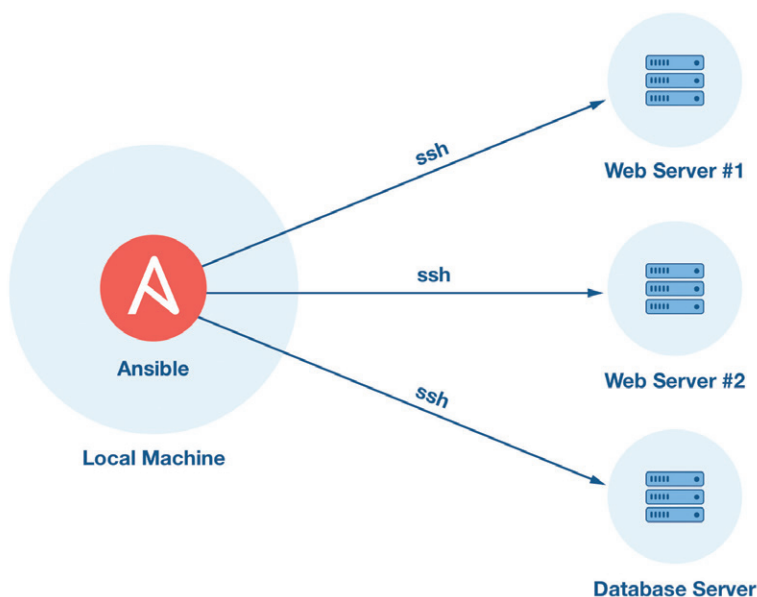
```
pip install ansible
```

To run *Ansible*, we need two things beside the software itself: a so-called inventory file telling *Ansible* which servers to execute the tasks on, and a playbook which contains the task to be executed. The inventory file follows the syntax of ini-files and contains hostnames, IP addresses and specific settings for each host. It is also used to cluster hosts into groups, which makes it easier to run tasks in only one group of servers (eg web servers and database servers). An example inventory file might look like this:

```
server1 ansible_host=127.0.0.101
server2 ansible_host=127.0.0.102 ansible_user=deploy
server3 ansible_host=127.0.0.103 ansible_port=30
[webservers]
server1
server2
[dbserver]
server3
```

In this example, we define three servers: the first one is called `server1` and has its IP address set to 127.0.0.101; for the second server, we define that *Ansible* should log in using the user **deploy**; the third server has a different SSH port set. There are many other settings that one can define here, and all of them are optional.

We also put the three servers into two groups, one called **webservers** and one called **dbserver**. These groups can be used to determine which tasks should



be run on each group of servers (eg only installing **nginx** on servers from the group **webserver**). Servers can be in more than one group and there are ways to define subgroups as well.

With this inventory file, we can start using *Ansible*. Let's try to ping those servers with a so-called *ad-hoc* task, which means we won't write any playbook files right now:

ansible all -i inventory -m ping

This tells *Ansible* to run the **ping** module on all hosts specified in the file named **inventory**. A module is an addition to the core *Ansible* software and offers ways to interact with common software. Instead of writing shell commands for every task, a module for eg **apt** offers a simple way to tell *Ansible* which software to install through **apt**.

If you run the command, you should get immediate feedback if everything is correct in your inventory file. If *Ansible* can't find your host, cannot SSH into it or isn't able to run the task after the connection is established, you need to revisit the inventory file and see if the correct options are set. If *Ansible* can't execute tasks, make sure Python 2 is installed on the remote machine and that the user *Ansible* uses to log in can use it.

These *ad-hoc* commands can be used to execute one-off commands on all servers without manually logging into them one at a time. It can be used to restart a specific process on all servers (or a group of servers), inspect log files or ask for status reports of various sorts.

Before we dive into playbooks and how to tell *Ansible* what tasks to run, let's figure out what steps we should perform in order to harden our server:

- For any server, the root account should have a good password and should not be used to work with directly. Instead, a different user which can perform root tasks using **sudo** should be used.
- This user should be able to log in with his SSH key instead of password.
- For SSH, we should only accept SSH keys to log in and only allow our user to log in.
- A firewall that locks down unused ports protects the server against software that might run but does not need to be accessed from the outside (like a database which is only accessed locally).
- The *fail2ban* software uses log files from the server

Ansible from source

As *Ansible* is written in Python, it can be installed from source pretty easy. Get the source code either as a download from the website or using *Git*, install the dependencies and run the setup script. The *Ansible docs* contain a tutorial on running *Ansible* from source.

The advantage is that you are able to run the latest version and you can update to beta versions if needed. You can also apply patches, which might already work but are not released yet to fix bugs you are experiencing. And of course, adding your own patches is possible as well.

```
$ ansible all -i inventory -m ping
homer | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Running the **ping** module to get "pong" back for every host in the inventory file.

to block IPs that try malicious stuff.

We'll go through those in a somewhat random order to explore *Ansible* step by step. The final playbook will have all tasks in a specific order that makes sense (eg creating a different user before locking down SSH access for root). This tutorial and the playbook will be targeting Ubuntu 14.04 servers, but with a little adjustment the playbook can be run on any server.

Let's start with setting up a playbook and using it to install *fail2ban*. This is what a basic playbook with one task which installs the *fail2ban* software through **apt** looks like:

To run Ansible we need two things: an inventory file and a playbook containing the tasks to be executed

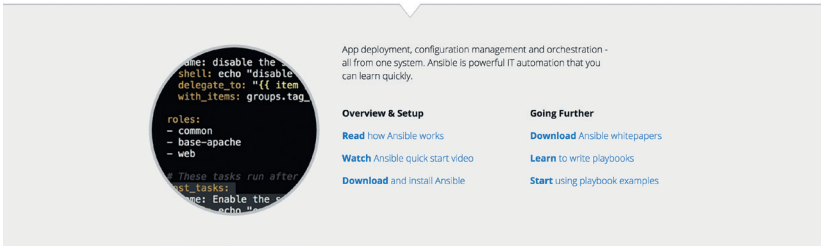
```
---
- hosts: all
  become: yes
  tasks:
  - name: install fail2ban
    apt:
      pkg=fail2ban
      state=latest
```

The syntax used is YAML, which is an easy to use "language" better suitable for humans than for example JSON. The **hosts** key holds the group of hosts for which the tasks should be applied. Using **all** here means that all hosts specified in the inventory file will be used. The **become** key tells *Ansible* that it should become the root user for the tasks run within this playbook. As we're going to run a lot of tasks which require **sudo**, which is easier than specifying it for every task.

Each task has a name that's used when printing the result to the shell. Below the name we specify the name of the module to be used. Each module has its own parameters depending on what it needs to perform the task at hand. For **apt**, we need to specify the package we want to install using the **pkg** parameter. The state can be either **absent**, which means it's uninstalled if currently installed, **present** to install it once and if it's installed do nothing or **latest**, which updates the package if there is a new version.

Ansible tasks should be idempotent, so instead of specifying what should be done tasks specify the state which should be achieved. In our first example,

STEP 1: LEARN HOW TO ANSIBLE



The *Ansible* website contains a lot of useful content to learn *Ansible*.

the state we want to achieve is that the **fail2ban** package is installed in the latest version. The module itself takes care of the steps needed to get there.

For **apt** to work correctly, we should also do an update:

```
- name: update apt cache
apt:
cache_valid_time=3600
update_cache=yes
```

Again, we are using the **apt** module, but this time we specify the **update_cache** parameter as well as **cache_valid_time** to prevent **apt** updating every time we run the playbook and instead skip this task if **apt** was updated in the last hour (3600 seconds).

To secure **sshd**, we should adjust the configuration file. We can use the **copy** module to copy a locally stored **sshd_config** file to our server. It's best to copy the current **sshd_config** from one of your servers using **rsync**:

```
rsync username@ip:/etc/ssh/sshd_config .
```

You should find the **sshd_config** file in your

Ansible has a lot for you, especially if you want to manage more servers or if the setup is getting more complex

current working directory. Make sure to change **PermitRootLogin** to **no** (and remove any **#** at the beginning of the line) and **PasswordAuthentication** to **no** as well.

This prevents the most common attack, which is trying to SSH into the server, using the root user and various passwords. The task to copy this file to every server would look like:

```
- name: change sshd config
copy:
dest=/etc/ssh/sshd_config
src=sshd_config
notify: restart sshd
```

This will take the file at **src** and copy it over to the server, storing it at **dest**. We could also use the **template** module, which would mean that *Ansible* would run the configuration file through the *Jinja2* templating engine. This lets you use variables inside the templates, which can be passed in from the playbook. The last line in our example specifies that if this task changes anything, a so-called handler is

notified. Handlers are used to restart services like **sshd** to make sure the changes to the config have any effect. We need to specify our handler not within the list of tasks but add another list with the key **handlers**:

```
handlers:
- name: restart sshd
service:
name=sshd
state=restarted
```

Ansible groups all notify events and execute them at the end of the playbook run, once for each handler. This means you can call notify many times for one handler but it only gets executed once.

We also said we needed a firewall to protect our server. Given the complexity of *iptables*, **ufw** is usually a better tool for the job. We need to install it, open up the ports we really need (22 for SSH, 80 for HTTP and maybe 443 for HTTPS) and enable **ufw**. We also need to restart **ufw** after this.

To run a task more than once with different arguments we can use **with_items**. *Ansible* runs the task for each item in the list and replaces **{{ item }}** with the current text. A task using **with_items** could look like this:

```
- name: allow various ports in ufw
ufw:
port={{ item }}
proto=tcp
rule=allow
state=enabled
with_items:
- 22
- 80
- 443
notify: restart ufw
```

Ansible executes this task three times, for "22", "80" as well as "443". Only the port changes, the other parameters stay the same. Afterwards, all three ports are allowed within **ufw**.

We didn't change the password for root yet. Even if root cannot SSH into the server any more, a strong password is still recommended. The **user** module has got us covered:

```
- name: change root password
user:
```

Different operating systems

Ansible does not provide any layer on top of modules, meaning that a playbook containing tasks to interact with **apt-get** will fail on a system that does not have **apt-get** available, like Red Hat-based Linux systems. There are a few ways to work around this which are out of the scope of this tutorial. The documentation offers some ideas on how to get started if you need this.

If you want to run *Ansible* from different host systems, you need to install *Ansible* on all of them. This can be challenging if Windows hosts are involved or if some hosts are running an old version of *Ansible*. You can use *Vagrant* or *Docker* to put *Ansible* into a controlled environment and run it from there.

My first 5/10 minutes on a server

This tutorial is loosely based on two blog posts named “My First 5 Minutes On A Server; Or, Essential Security for Linux Servers” (<https://plusbryan.com/my-first-5-minutes-on-a-server-or-essential-security-for-linux-servers>) and “My First 10 Minutes On a Server – Primer for Securing Ubuntu” (www.codelitt.com/blog/my-first-10-minutes-on-a-server-primer-for-securing-ubuntu). Both tutorials contain additional steps which were not included in this tutorial in order to keep it short. The reader can go ahead and implement the missing steps using *Ansible* as shown above.

A word of caution: All these steps are basic first steps and do not protect you against sophisticated attacks as well as exploits of bugs in software you are using. While these steps will prevent the most common attacks, hackers might still be able to get into your system.

```
name=root
```

```
password=encryptet_password
```

```
state=present
```

The value for the password cannot be plain text, as this would be unsecure, given that playbooks are plain-text on your hard drive and usually live inside a Git repository as well. To generate the encrypted password, use the

```
mkpasswd --method=SHA-512
```

utility or refer to the *Ansible* FAQ at <https://docs.ansible.com/ansible/faq.html#how-do-i-generate-encrypted-passwords-for-the-user-module> for other ways to generate the password.

To create a new user, the same module and parameters can be used. Adding the parameter **groups** and setting it to **sudo** as well as adding the parameter **bash** set to **/bin/bash** makes sure the user is created with sensitive settings. Your SSH key for logging in through SSH can be copied from `~/.ssh/id_rsa.pub` to `~/.ssh/authorized_keys` on the remote server.

Allowing the **sudo** group to run **sudo** without a password can be done by using the **lineinfile** module:

```
- name: make sudo group user access sudo without password
```

```
lineinfile:
```

```
dest=/etc/sudoers
```

```
regexp="%sudo"
```

```
line="%sudo ALL=(ALL:ALL) NOPASSWD:ALL"
```

Here we specify a regex which is used to find the line we want to replace with what we specified as the **line** argument. In our case, we are looking for a line that starts with **%sudo**. If this line is found, it is replaced with our line. If it's not found, the line is appended to the bottom.

With the full setup in place, we can run the playbook against our servers:

```
ansible-playbook -i inventory tasks.yml
```

We use the

```
ansible-playbook
```

command and pass the inventory file we created as well as the playbook containing the tasks we want to run. *Ansible* shows us, for each task, on which server it

```
PLAY [homer] *****
TASK [setup] *****
ok: [homer]
TASK [update apt cache] *****
ok: [homer]
TASK [upgrade system using apt] *****
changed: [homer]
TASK [install apt packages] *****
ok: [homer] => (item=[u'fail2ban', u'rsync', u'sudo'])
TASK [make group user access sudo without password] *****
ok: [homer]
TASK [change ssh config] *****
ok: [homer] => (item={u'regex': u'^#?Port', u'line': u'Port 30'})
ok: [homer] => (item={u'regex': u'^#?LoginGraceTime', u'line': u'LoginGraceTime 20'})
ok: [homer] => (item={u'regex': u'^#?PermitRootLogin', u'line': u'PermitRootLogin no'})
ok: [homer] => (item={u'regex': u'^#?PasswordAuthentication', u'line': u'PasswordAuthentication no'})
ok: [homer] => (item={u'regex': u'^#?DebianBanner', u'line': u'DebianBanner no'})
ok: [homer] => (item={u'regex': u'^#?MaxAuthTries', u'line': u'MaxAuthTries 2'})
ok: [homer] => (item={u'regex': u'^#?MaxSessions', u'line': u'MaxSessions 2'})
TASK [Set correct timezone] *****
```

was executed and if the task changed something on the remote server, if it failed and so on.

The full playbook, containing all tasks outlined above, together with the inventory file and an example **sshd_config** file can be found at <https://gist.github.com/Sgoettschkes/f737a89b0481f741a39e1943ce2dcd9b>.

Ansible has a lot more for you, especially if you want to manage more servers or if the setup is getting more complex: variables can be used almost anywhere in *Ansible* playbooks. They can be defined on many levels (within a playbook, for each group or host and even on the command line when running the *Ansible* playbook). This way, you can create flexible playbooks that can be filled in with variables depending on which host the task is executed on.

So-called facts are also variables that *Ansible* fills with information from the servers against which you are currently running the playbook. This information includes the IP address, amount of RAM, network information and much more. This way, you could set up a **ufw** firewall to allow access to your database servers only from the web servers in a dynamic way, reading the IP addresses from all servers in the group web server and using these values to set up **ufw** allow rules. If you change the inventory file later and run the playbook again, **ufw** gets adjusted automatically.

Splitting up your playbook into roles you can reuse your playbooks in different scenarios. One role could be SSH and it could take care of **sshd_config**. This role can then be used in many projects and you only need to maintain it once. There is even a public repository for roles, called Ansible Galaxy: <https://galaxy.ansible.com>.

Using *Ansible* is a great way to not spend much time doing infrastructure work and knowing exactly what happened to each server. With the playbook outlined above, your servers will be more secure and easy to set up in the future. And it makes a good base to add your own setup on top! 📖

Sebastian Göttches is interested in Python, Dart, Ansible, PHP, and a little thing called the Google Cloud Platform.

Output from running an *Ansible* playbook against one remote server.

PROCESS AND VISUALISE APACHE LOGS FILES

Use AWK and R to reveal useful information from your Apache log files.

MIHALIS
TSOUKALOS

Why do this?

- Process Apache log files using AWK
- Visualise the information found in Apache logs
- Identify possible security threats

PRO TIP

Apache log files are usually very large, especially if you have a popular website; the trick is to test your scripts using smaller log files before trying to process the actual log files. However, it is considered a bad practice to experiment on production machines, so it's better to transfer your log files on another machine.

There are two main reasons for working with log files: the first one is for creating reports and summaries; and the second one is for discovering abnormal events, which can either be possible security threats or indications of a misbehaving executable.

This tutorial will tell you how to process log files that come from an *Apache* web server in order to extract useful information using traditional Unix command line tools such as *Grep* and *Awk*. It will also show you how to visualise the extracted information, because a plot can help you focus on what really matters. Additionally, non-technical people feel more comfortable with plots than with text reports. We're going to use the *R* statistical package for data visualisation, but you can also use the Julia programming language, *Gnuplot*, *Matlab*, Python, Perl, etc. Although scripting languages such as Perl, Python and Ruby can be used for processing log files, nothing beats the speed of *Awk* for relatively simple jobs. Scripting languages are more suitable for advanced things

such as inserting data into a database or turning data into a beautiful PDF report.

The good thing with all presented command line utilities is that none of them changes its input, which means that you can even work with active *Apache* log files without the fear of changing or destroying them.

The format of an Apache log entry

A common log entry in an *Apache* log file will be similar to the following:

```
73.208.169.12 - - [28/Jun/2016:10:20:02 +0300] "GET /misc/menu-leaf.png HTTP/1.1" 200 457 "http://www.mtsoukalos.eu/Enable-Root-user-on-El-Capitan" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/601.6.17 (KHTML, like Gecko) Version/9.1.1 Safari/601.6.17" 604
```

However, this is not the only format that you might come up with, because *Apache* enables you to define your own. The right place for this is inside **apache2.conf**, using lines beginning with **LogFormat**:

```
$ grep -i logformat /etc/apache2/apache2.conf
```

```
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" " vhost_combined
```

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" " combined
```

```
LogFormat "%h %l %u %t \"%r\" %>s %O" common
```

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" %D" myformat
```

The name of each log format is defined by the last word of each line (**vhost_combined**, **combined**, **common** and **myformat**). You can find more about the various formatters used for creating custom log formats at http://httpd.apache.org/docs/2.4/mod/mod_log_config.html.

Should you wish to use any one of them, you should write the following entry inside the definition file of a website, which most often is a **VirtualHost** block:

```
CustomLog /srv/www/a_site/logs/access.log myformat
```

The rest of the tutorial will use log files that use the **myformat** format.

The sed utility

sed is another handy command line utility for processing text input. Imagine that you have log entries that format dates and times as follows:

```
[27/Jun/2016:22:49:03 +0300]
```

```
2. mtsouk@mail: /srv/www/www.mtsoukalos.eu/logs (ssh)
log$ cat access.log | awk {'print $8'} | sort | uniq -c
 486 HTTP/1.0"
 4167 HTTP/1.1"
log$ cat access.log | awk {'print $9'} | sort | uniq -c | sort -nr
 4402 200
  134 403
   19 404
   13 304
    5 302
log$ zcat access.log.*.gz | awk {'print $4'} | awk -F/ {'print $1'} | sort | uniq -c | awk {'print $2 "\t" $1'} | sed 's/^\[//g'
 01 26829
 02 24562
 03 24722
 04 26451
 05 22769
 06 24255
 07 29634
 08 27316
 09 28643
 10 27131
 11 27398
 12 26459
 13 26870
 14 27949
 15 28090
 16 25917
 17 27909
 18 28378
 19 25635
 20 25071
 21 24233
 22 25144
 23 25536
 24 22767
 25 23238
 26 26455
 27 24651
 28 24348
 29 26992
 30 23292
 31 13175
log$ zcat access.log.*.gz | wc
791801 17205880 197700573
log$ wc access.log
 4576  92034 1078247 access.log
log$
```

Some of the text reports you can get when processing *Apache* log files with *Awk*.

The following **sed** command replaces Jun with June on a log file:

```
$ sed 's/-Jun-/-June-/g' access.log
```

The next **sed** command prints the first 1000 lines of a file:

```
$ sed -n 1,1000p access.log
```

The next command tells **sed** to print just the first line of the input:

```
$ sed '1!d' access.log
```

The last example shows how to globally delete a single character, in this case **[**, from a text stream using **sed**:

```
$ cat text.file | sed 's/\[/g'
```

Sed is priceless when you want to clean up log files from control characters or perform global search and replace operations, but it cannot do very complex things because it is not a programming language.

Using Awk

Awk is a handy programming language for processing text data with many capabilities. As an example, you can find out the IPs of your top five clients with the following command:

```
$ cat access.log | awk {'print $1'} | sort | uniq -c | sort -nr | head -5
```

Although this tutorial is not about Awk, it is worth explaining the previous command. The **cat** command gives the contents of **access.log** as input to the **awk** command, which prints the first field of each line that is the IP address of the client. The **sort** command sorts all lines and the **uniq -c** command both deletes repeated lines and precedes each output line with the count of the number of times the line occurred in the input. Then you sort the output numerically in reverse order before printing the first five lines.

Similarly, you can create a summary of all HTTP status codes:

```
$ cat access.log | awk {'print $9'} | sort | uniq -c | sort -nr
```

A very handy report is the one that counts the number of connections per hour of the day:

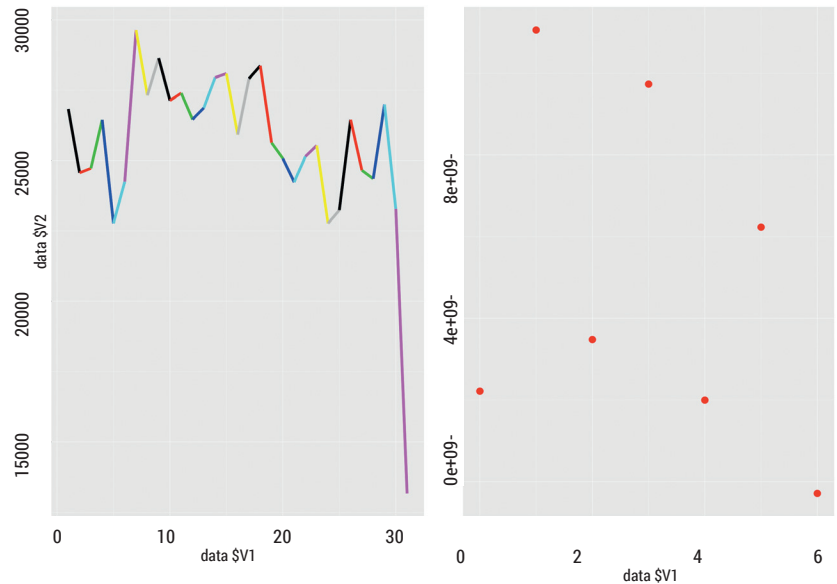
```
$ cat access.log | awk {'print $4'} | awk -F: {' print $2 '} | sort | uniq -c | awk {' print $2 "\t" $1 }
```

As a day has 24 hours, you will get at most 24 lines of output. The following Awk script, saved as **bytesDOW.awk**, calculates the number of bytes

About Awk, sed and grep

Awk is an interpreted programming language designed for easy, productive and fast text processing, data extraction and reporting created at Bell Labs back in 1970s; Linux systems use the much improved GNU Awk version.

Sed (Stream Editor) is a tool for performing global search and replace operations on text files, whereas *Grep* is a command line tool for searching text files using regular expressions. The *Grep* tool cannot change its input, which is the main reason for using utilities such as Awk and *Sed*. Both Awk and *Sed* can replace *Grep*, but its simplicity makes *Grep* unique. There exist many *Grep* variations including *pgrep*, *egrep* and *fgrep*.

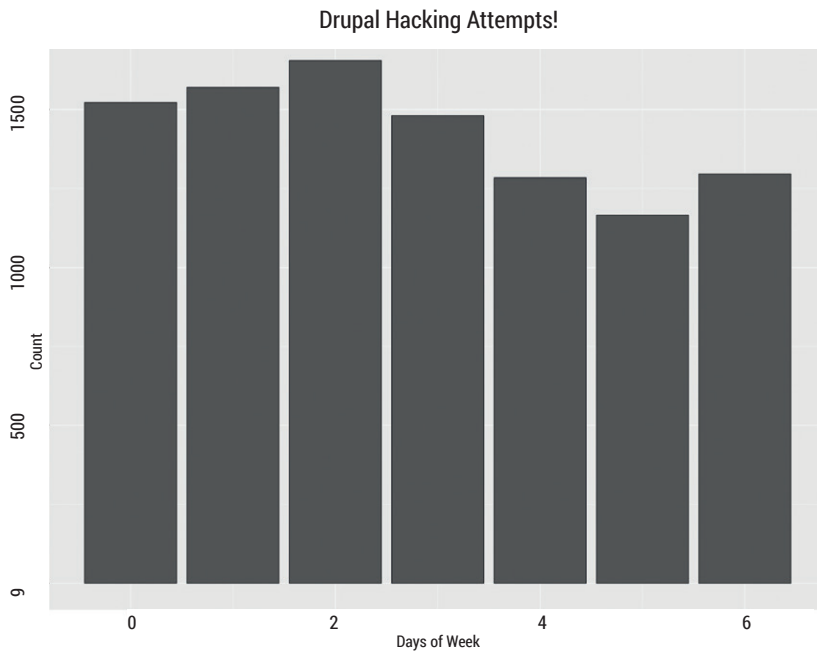


served by the web server per day of the week:

```
# 0 is Sunday, etc.
function dayOfWeek(year, month, day) {
    day_of_week = 0;
    if (month <= 2)
    {
        month += 12;
        year--;
    }
    day_of_week = (day + month * 2 + int(((month + 1) * 6) / 10) + year + int(year / 4) - int(year / 100) + int(year / 400) + 2);
    day_of_week = day_of_week % 7;
    return ((day_of_week ? day_of_week : 7) - 1);
}
BEGIN {
    month["Jan"] = 1;   month["Feb"] = 2;
    month["Mar"] = 3;
    month["Apr"] = 4;   month["May"] = 5;
    month["Jun"] = 6;
    month["Jul"] = 7;   month["Aug"] = 8;
    month["Sep"] = 9;
    month["Oct"] = 10;  month["Nov"] = 11;
    month["Dec"] = 12;
}
{
    split($4, left, ":");
    split(left[1], desired, "/");
    year = desired[3];
    myMonth = month[desired[2]];
    day = substr(desired[1], 2);
    currentDay = dayOfWeek(year, myMonth, day);
    myData[currentDay] += $10;
}
END {
    for (val in myData)
        print val, "\t", myData[val];
}
```

As a week has seven days, the output will have seven lines. It is necessary to know which field holds

This is the output of the **plotData.R** script, which plots data acquired by processing *Apache* log files using Awk.



This is the output of the **drupal.R** script that reads a file named **drupal.data** and plots it.

the number of bytes transferred – in this case it is the 10th field.

You can also create a report that shows the number of connections per day of the week using a similar Awk script, which is saved as **connectionsDOW.awk**. The following output shows the differences between **bytesDOW.awk** and **connectionsDOW.awk** using the **diff** utility:

```
$ diff connectionsDOW.awk bytesDOW.awk
32c32
< myData[currentDay]++;
```

The first step to any successful data visualisation is knowing what kind of information you're looking for

```
---
```

```
myData[currentDay] += $10;
```

The way Awk finds out the day of the week needs some explanation: as the name of the day cannot be found in the log file, we will have to find it on our own. An Awk function named **dayOfWeek** does all the job for us – as you can see, finding the day of the week from a date is not a trivial task. Additionally, you also need to convert the name of the month to a number, which is done with the help of an associative array named **month**.

Both **connectionsDOW.awk** and **bytesDOW.awk** must be executed as follows:

```
$ cat ./access.log | awk -f connectionsDOW.awk
$ cat ./access.log | awk -f bytesDOW.awk
```

The output of **connectionsDOW.awk** will be used by R and therefore it is saved as **connectionsPerDayOfWeek.data**.

The next report shows the number of connections per day of the month, which gives you a pretty good idea of what is going on on your website:

```
$ cat access.log | awk '{print $4}' | awk -F/ {'print $1'} | sort | uniq -c | awk '{print $2 "\t" $1}' | sed 's/\\/g'
```

The output from the previous command will be used by R later on in this tutorial, so it is saved as **dayOfMonth.data**. Note that the two fields of each line are separated by a tab character.

You can easily create analogous reports that show the total number of connections per month, per year, per IP address, per web page, etc.

The following code prints a report about the various versions of the HTTP protocol:

```
$ cat access.log | awk '{print $8}' | sort | uniq -c
406 HTTP/1.0"
3902 HTTP/1.1"
```

The last report that will be created will show the number of unique IPs per day of the month:

```
$ cat access.log | awk '{print $1, $4}' | awk -F/ {'print $1'} | awk -F\ {'print $2, $1'} | sort | uniq | awk '{print $1}' | uniq -c | awk '{print $2, $1}'
```

Using the Grep utility

The following **Grep** command finds all log entries that contain a given IP address, and counts them using **wc**:

```
$ grep 66.249.64.139 ./access.log | wc
30 558 6835
```

The next **Grep** command shows all entries that both contain a given IP address and return a 404 status code:

```
$ grep 66.249.64.139 /srv/www/www.mtsoukalos.eu/logs/access.log | grep " 404 "
```

The last example finds all log entries that contain files with the **.png** or **.PNG** extension:

```
$ grep '\.png\\\.PNG' ./access.log | wc
1157 26062 308592
```

A variation of the previous command finds all accesses to ZIP files:

```
$ grep '\.zip\\\.ZIP' ./access.log | wc
2 48 608
```

Please bear in mind that **Grep** does not allow you to add logic to your searches; therefore, doing advanced operations such as comparisons requires tools like Awk, Perl or another scripting language.

Using R for visualisation

R offers a plethora of plots and graphs that you can use; it's your job to select the right kind of plot for presenting your data. It is now time to visualise both **connectionsPerDayOfWeek.data** and **dayOfMonth.data** using R.

The following R script, which is named **plotData.R**, will plot both data files:

```
#!/usr/bin/env Rscript
require(ggplot2)
data <- read.table("./connectionsPerDayOfWeek.data",
header=FALSE)
outputfile <- paste("cDay", ".png", sep="")
```

```
png(filename=outputfile, width=1200, height=1600)
p <- ggplot(data, aes(data$V1, data$V2)) + geom_
point(size=10, colour="red")
print(p)
data <- read.table("./dayOfMonth.data", header=FALSE)
outputfile <- paste("cDayOfMonth", ".png", sep="")
png(filename=outputfile, width=1200, height=1600)
q <- ggplot(data, aes(data$V1, data$V2)) + geom_
line(size=4, colour=data$V1)
print(q)
```

You can execute the R script as follows:

```
$ chmod 755 time.R
$ ./time.R
```

The script will produce two PNG files named **cDay.png** and **cDayOfMonth.png** as defined in the code.

Visualising for security

The first step to a successful visualisation is knowing what kind of information you're looking for, which means that you should also know how your web application works. For a Drupal site you can start by monitoring the **GET /?q=node/add HTTP/1.1**, **GET /?q=user/register HTTP/1.1**, **GET /?q=node/add HTTP/1.0** and **GET /?q=user/register HTTP/1.0** requests that indicate direct hack attempts. The easiest way to get the log entries that contain such requests is with the help of *Grep*. Then, you will process the log entries using **connectionsDOW.awk** and save the data in a file named **drupal.data**:

```
$ grep "GET /?q=node/add HTTP/1.1" \| "GET /?q=user/
register HTTP/1.1" \| "GET /?q=node/add HTTP/1.0" \|
"GET /?q=user/register HTTP/1.0" access.log | awk -f
connectionsDOW.awk > drupal.data
```

Last, you will process **drupal.data** using the following R script:

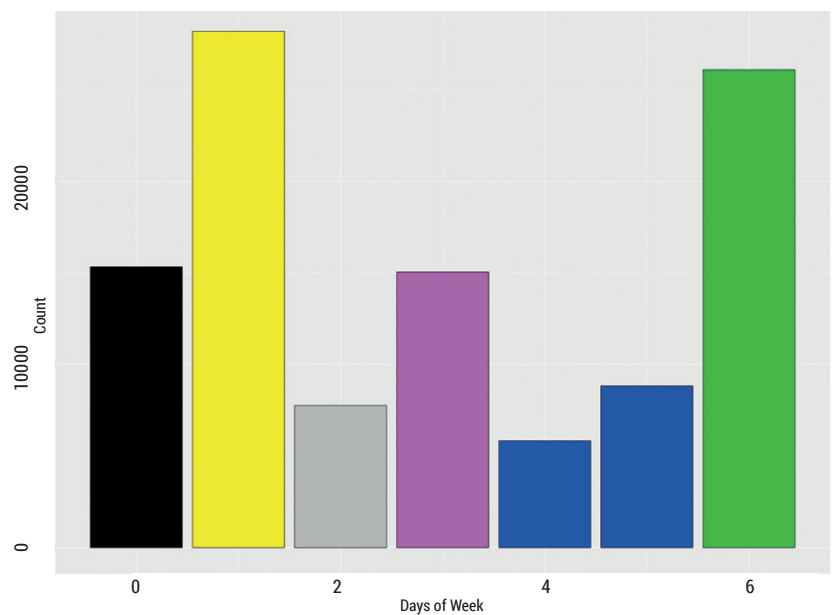
```
#!/usr/bin/env Rscript
require(ggplot2)
data <- read.table("./drupal.data", header=FALSE)
outputfile <- paste("drupal", ".png", sep="")
png(filename=outputfile, width=1600, height=1200)
ggplot(data, aes(x = data$V1, y = data$V2)) + geom_
bar(stat = "identity", colour="black") + ggtitle("Drupal
Hacking Attempts!") + labs(x="Day of Week", y="Count")
```

Why visualise?

All web server administrators understand the importance of having a high-level view of their web traffic. Plots and graphs enable you to have a quick overview of your web server traffic, which is very difficult to watch otherwise because web traffic is getting bigger and bigger.

However, do not forget that at the end of the day what really matters is the actual data! In other words, even the most impressive plot or graph cannot save you from data that cannot reveal the truth, so, do not try to measure the performance of a web server during weekends or do not search for hacking attempts during normal working hours, because web servers have less traffic during weekends and hacking attempts usually take place late at night or during holidays; use your common sense!

Joomla Hacking Attempts!



The output of **joomla.R** shows the number of hacking attempts per day of the week (Sunday = 0, etc.).

The output is saved as **drupal.png**.

The administration page of a Joomla site can be found at the **/administrator/** path, which means that only an administrator has the right to visit this URL. So, the first step is using *Grep* is to find out all paths that contain **/administrator** or **/administrator/** and then process the output using the **connectionsDOW.awk** Awk script:

```
$ cat ./access.log | grep -i administrator | awk -f
connectionsDOW.awk > joomla.data
```

The next R script plots the collected information:

```
#!/usr/bin/env Rscript
require(ggplot2)
data <- read.table("./joomla.data", header=FALSE)
outputfile <- paste("joomla", ".png", sep="")
png(filename=outputfile, width=1600, height=1200)
p <- ggplot(data, aes(x = data$V1, y = data$V2)) + geom_
bar(stat = "identity", fill = data$V2, colour="black")
p <- p + ggtitle("Joomla Hacking
Attempts!") + labs(x="Day of Week",
y="Count")
p <- p + theme(plot.title = element_
text(size = rel(3), colour = "black"))
print(p)
```

The generated plot is saved as

joomla.png.

As you can understand, the format of the log file is not important; what is really important is recognising the data that matters and processing it. Additionally, as long as you extract your data in a standard format, no other changes need to be made to the rest of the code. 📄

Mihalís Tsoukalos is a Unix administrator, mathematician and programmer who enjoys writing technical articles.

PRO TIP

The cron tool is your friend because it enables you to execute your scripts at night or during weekends. In other words, do not execute heavy scripts when a Linux machine has more important things to do, like serving user requests.

ELASTIC BEANSTALK: DEPLOY A WEB APP

Host and scale a Golang web application on Amazon Elastic Beanstalk.

AMIT SAHA

Why do this?

- Get started with Platform as a Service (PaaS)
- Automatically scale your latest web deployment
- Add another buzzword to your CV

Amazon Elastic Beanstalk is a Platform as a service (PaaS) and is part of Amazon Web Services (AWS). Basically, this means that you outsource your worry of managing servers including the operating system that is running on it to AWS. All you need to think about now is writing your web application; as long as this starts correctly, Amazon Elastic Beanstalk will take care of the rest. We get the advantage of auto scaling – automatically increasing and decreasing the number of instances of our application based on network traffic (for example), and can upgrade to new versions of our application without downtime, as well as access to a whole bunch of other AWS services. The trade-off to using a PaaS is, of course, that we give away our control over the software (including the operating system) that runs on it.

In this article, we will write a simple web application in Golang and deploy it to Elastic Beanstalk. The final web application we will write is an Integer Obfuscation service: pass an integer to it, and you will get a JSON object with a random string back. An example request and response using the `curl` command looks as follows:

```
$ curl http://linux-voice-5.us-west-2.elasticbeanstalk.com/?id=1
{"id":1,"obfuscated_id":"6d6p4M"}
```

The first step to set up Golang on our system is to install the Go compiler and other tools. You may

choose to install the Go tools using your distro's package manager. However, it's likely that it may be lagging behind the upstream release. Hence, we will install it manually. First download the 1.6 Linux binary from <https://golang.org/dl/>; then untar the package with `sudo tar -C /usr/local -xzf go1.6.2.linux-amd64.tar.gz`. Next, add `/usr/local/go/bin` to your PATH by adding `export PATH=$PATH:/usr/local/go/bin` in your `.bashrc`.

The next step is to set up our Go workspace – a directory where all our Golang source code will live and an environment variable `GOPATH` whose value is this directory. My workspace is set up as the directory `golang` in the directory `$HOME/work` (for the purpose of this article, my `$HOME` is `/home/vagrant`):

```
$ mkdir $HOME/work/golang
```

In this directory, we will create a `src` sub-directory:

```
$ mkdir $HOME/work/golang/src
```

Next, in our `.bashrc` file or similar, we will add the following:

```
export GOPATH=$HOME/work/golang
```

Now, if we open a new terminal session, the command `go env GOPATH` should print the above path.

Our web application

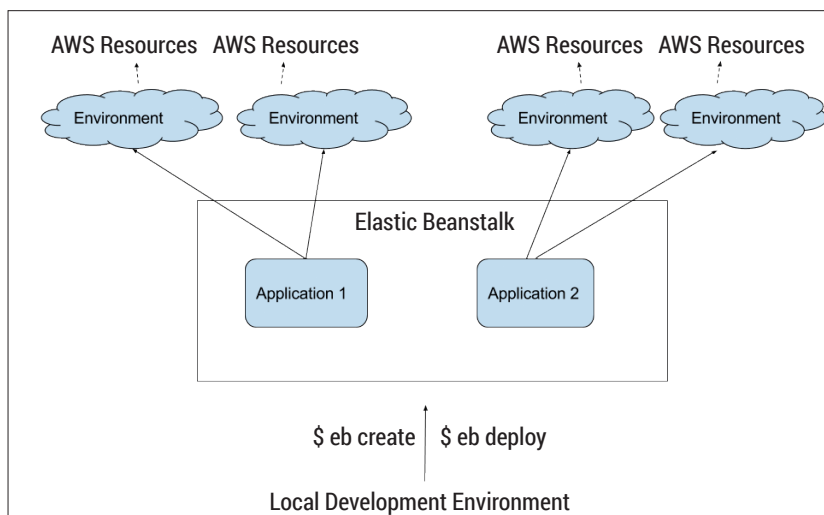
Now, we will write our web application. The current stable version of the language is 1.6, with 1.7 due out soon. But Elastic Beanstalk supports only 1.3 and 1.4 now. Hence for our web application, we are left with two choices:

- Don't use any Golang feature which is not present in 1.4.
- Deploy our web app as a docker container using the latest Golang features.

We will keep it simple first and attempt the first option, and then adopt the second deployment option. The next code listing shows our web application:

```
// Basic HTTP server listening on 5000
package main
import (
    "net/http"
    "fmt"
)
func handleRequest(w http.ResponseWriter, r *http.Request) {
```

Figure 1: Applications hosted on Elastic Beanstalk. AWS Resources include the Linux VM instances, Auto Scaling groups, Elastic Load Balancer, Security groups and others.



```
fmt.Fprintf(w, "Hello World")
}
func main() {
    http.HandleFunc("/", handleRequest)
    http.ListenAndServe(":5000", nil)
}
```

The first line in the above program is a comment. Next, we declare the package for our program. We declare that this program is an executable by specifying the package as **main**. Next, we import two packages from the standard library **net/http** and **fmt**. Then we write a function, **handleRequest()**, which handles any request sent to our web application. The first parameter is a variable, **w** of type **http.ResponseWriter**, which corresponds to the read end of the client. Anything we write to this object forms our response to the client's request. The second parameter, **r** is a variable of type **http.Request** and gives us access to the client request we are serving. This includes the request type, the request body, headers and others. In this function, we write a string "Hello World" to **w**, and this any client connecting to our web application will get "Hello World" back.

Next, we have the **main()** function, which is where our program's execution starts. The first statement tells us that any request made to the root path **"/** of our web application should be sent to the **handleRequest()** function for processing. We do so using the **HandleFunc()** function in the **http** package. To actually start the HTTP server, we call the **ListenAndServe()** function with the first argument being the address to listen on. **:5000** means to listen on port 5000 on all interfaces. The reason we use this port is because that's where Elastic Beanstalk expects our web application to listen on. Save the above program in a file **\$GOPATH/src/github.com/amitsaha/linux_voice_5/application.go**, navigate to the directory **\$GOPATH/src/github.com/amitsaha/linux_voice_5** and then run it using **go run application.go**:

```
$ go run application.go
```

Our web application is now running on port 5000. If we send a **curl** request from another terminal, we should get a "Hello World" message back:

```
$ go run application.go
```

Our web application is now running on port 5000. If we send a **curl** request from another terminal, we should get a "Hello World" message back:

```
$ curl 127.0.0.1:5000
```

```
Hello World
```

Now that our first web application is ready, let's host it on Elastic Beanstalk. Use **Ctrl + C** in the terminal where you started the server to stop the server. First, we will initialise a Git repository for our web application code:

```
$ pwd
```

```
/home/vagrant/work/golang/src/github.com/amitsaha/linux_voice_5
```

```
$ git init
```

```
Initialized empty Git repository in /home/vagrant/work/golang/src/github.com/amitsaha/linux_voice_5/.git/
```

Let's add the current **application.go** file and commit it:

```
$ git add application.go
```

```
$ git commit -m "First application version"
```

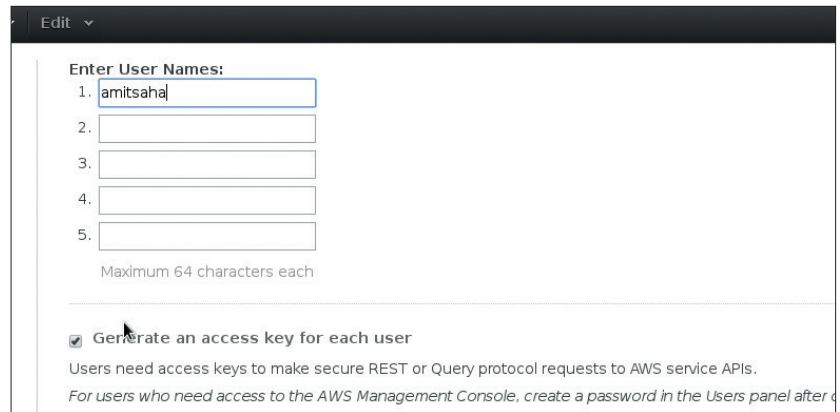


Figure 2: Creating a user to access AWS services

The first step for us is to create an Amazon AWS account by going to **https://aws.amazon.com/free**. While creating the account, we will be asked to supply a credit card information. There is a 12-month introductory offer where we will not be charged if our usage of AWS resources do not exceed the limits of the free tier. On the account creation page, you can read the various terms and conditions associated with the free tier, the various services and usage limits and this is also a good chance to have a quick read of all the various services that AWS provides.

Signing up for an Amazon AWS account

The user account that you just created is the "root" AWS account. Although we can start using AWS using this account, we don't recommend you do so: instead, we'll create what is referred to as an AWS IAM user by going to **https://console.aws.amazon.com/iam/home** and clicking on "Create New Users" (Figure 2).

Next, when prompted, download the credentials file (Figure 3). This is a CSV file having your username, AWS access key ID and secret access key.

Next, we need to give the user permissions for performing all the operations we need for working with Elastic Beanstalk. From the **https://console.aws.amazon.com/iam/home?#users** page, click on the username we created above, go the "Permissions" tab, then click on "Attach Policy". From the list of policies check "IAMFullAccess" and "AWSElasticBeanstalkFullAccess" and click "Attach Policy". Now, the "Permissions" tab for the user should show these policies (Figure 4).

Setting up Elastic Beanstalk CLI

The **awsebcli** Python package provides a command line interface to Elastic Beanstalk. It is a good idea to install it in its own virtual environment. If you do not have **virtualenv** installed, you can install it using your distro's package manager (on Ubuntu 14.04 and Fedora 23, you can install the package **python-virtualenv**). Next, create a virtual environment in a location of your choice. I will create a virtual environment in the **~/work/venvs** directory:

```
$ virtualenv ~/work/venvs/ebcli
```

```
$ . ~/work/venvs/ebcli/bin/activate
```

Once we have created the virtual environment and

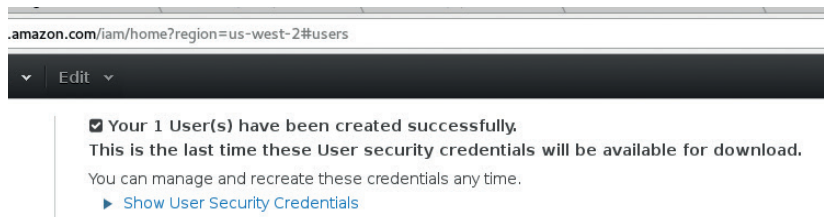


Figure 3: Download credentials file.

activated it, we can install the **awsebcli** package using **pip install awsebcli**. Let's verify the install:

```
$ eb --version
```

```
EB CLI 3.7.6 (Python 2.7.1)
```

At this stage the program **eb** is installed and ready to use. The first step is to create a configuration file where we will put our AWS credentials. Create a file **~/.aws/config** and fill in the following:

```
[profile eb-cli]
```

```
aws_access_key_id = <YOUR ACCESS KEY>
```

```
aws_secret_access_key = <YOUR SECRET ACCESS KEY >
```

The access key and the secret key are both available in the credentials file we downloaded when creating the account for the new user. Next, as the first step for deployment, we will initialise our Elastic Beanstalk application. Navigate to the directory where our code lives and run **eb init**:

We first select the AWS region where we plan to host our application. The region you choose would be determined by various factors including geographical closeness to your application's users', cost, compliance and the availability of the service in that region. For our case, we will simply select the default (**us-west-2**) region.

Next, we enter a name for our application. By default **eb init** suggests us to use the same as the current directory name. In the next two steps, we select the platform and the platform version (Go and Go 1.4 respectively). When using the Go 1.4 platform, Elastic Beanstalk expects the main program to be named as **application.go**, which is why we used that specific filename earlier. We answer **n** when asked if we want to set up SSH for our instance.

Based on our answers above, a configuration file **config.yml** is created in the directory **.elasticbeanstalk**. If you see the **.gitignore** file, you will see that the Elastic Beanstalk configuration file is not version controlled. This means that the configuration file will not be uploaded when our web application is uploaded to AWS or if we push our code to a remote repository.

The next step is create the environment for our web application. An "environment" encompasses your application code and all the AWS resources associated with the running instance of your application. These resources include EC2 instances, load balancer, an auto scaling group, S3 buckets, Security Groups and IAM roles. To create an environment, we use the **eb create** command:

```
$ eb create
```

You can look at all the AWS resources that our web application is currently using by signing in on the AWS

console and going to the individual AWS service's page for the same region as your web application.

At this stage, our application is deployed. Execute the command **eb open** while in the same directory and you should see "Hello World" on a web page. Congratulations! Your web application is now deployed. Let's go back to the two pieces of information **eb create** asked us for earlier:

```
Enter Environment Name
```

```
(default is linux-voice-5-dev):
```

```
Enter DNS CNAME prefix
```

```
(default is linux-voice-5-dev):
```

First, we are asked to specify the environment name. We can deploy your application into multiple environments running the same or different versions of our application. Later on, we will create a new environment that will be our production environment and another to act as our staging environment.

Next, we are asked to specify a DNS CNAME prefix for our web application. The default is the environment name, which means our application will be available at **http://linux-voice-5-dev.us-west-2.elasticbeanstalk.com**. I should mention here that you can use a custom domain for your application using AWS's Route53 services.

The "service role" that is created for our environment is how we give Elastic Beanstalk permission to carry out the various operations on our behalf. This includes creating and destroying AWS resources associated with our environment.

Inspecting your environment state

The **status** sub-command displays the status of an environment. If we don't specify an environment name, it displays the status of the current environment. The **eb logs** command displays the last 100 lines of various log files from the instance running your application. This includes access logs, error logs and activity logs. If we get any unexpected behaviour from your web application, these are the logs we will look into. To download the entire logs, we have to specify the **--all** switch:

The **events** sub-command can be used to retrieve a list of recent events – essentially a high-level summarised view of what has been happening in your current environment. When this is used with the **-f** flag, we can follow events as they happen:

```
$ eb events -f
```

```
INFO: createEnvironment is starting.
```

```
INFO: Using elasticbeanstalk-us-west-2-367082021788 as Amazon S3 storage bucket for environment data.
```

```
INFO: Created security group named: sg-db62cddb
```

```
INFO: Environment health has transitioned to Pending. Initialization in progress (running for 22 seconds). There are no instances.
```

```
INFO: Created load balancer named: awseb-e-2-AWSEBLoa-FUF7TCRQW3H4
```

```
INFO: Created security group named: awseb-e-23852cmkuv-stack-AWSEBSecurityGroup-RHBOGD32SNE9
```


INFO: Created Auto Scaling launch configuration named: awseb-e-23852cmkuv-stack-AWSEBAutoScalingLaunch Configuration-16GFEWY8H7O53

INFO: Created Auto Scaling group named: awseb-e-23852cmkuv-stack-AWSEBAutoScalingGroup-1SKOJXCY6RFJN

INFO: Waiting for EC2 instances to launch. This may take a few minutes.

...

We can specify a different environment name to the command as **eb events <environment-name>**.

Deploying our web app as a Docker container

Docker (www.docker.com) is a software containerisation solution that enables us to run the software we want to. This means that we can run our own Linux distribution using our own version of the software we want to. Hence, using Docker we can use a more recent version of Golang for our web application.

The instructions for installing the Docker engine on Linux are available at <https://docs.docker.com/engine/installation/linux>.

Once you have Docker engine installed, let's write the Dockerfile that Elastic Beanstalk will use to deploy our application. Save the following into a file **Dockerfile** in the same directory as our web application:

```
FROM golang
ADD application.go /go/src/github.com/amitsaha/linux_voice_5/application.go
EXPOSE 5000
CMD ["go", "run", "/go/src/github.com/amitsaha/linux_voice_5/application.go"]
```

The first statement in the Dockerfile states the base image on which we will deploy our application. The **golang** image (https://hub.docker.com/_/golang) is based on Debian and at the time of writing has Go 1.6 installed. Next, we copy our web application source, **application.go**, to a path **/go/src/github.com/amitsaha/linux_voice_5/application.go** in the image we will build. The statement **EXPOSE 5000** makes our web application accessible from the host operating system, and the final statement specifies the command we want to run when the image is run.

Before we go ahead with trying our our Dockerfile, let's first update our **application.go** file to the following:

```
// Basic HTTP server listening on 5000
package main
import (
    "fmt"
    "net/http"
    "runtime"
)
func handleRequest(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello World. Running: ")
    fmt.Fprintf(w, runtime.Version())
}
```

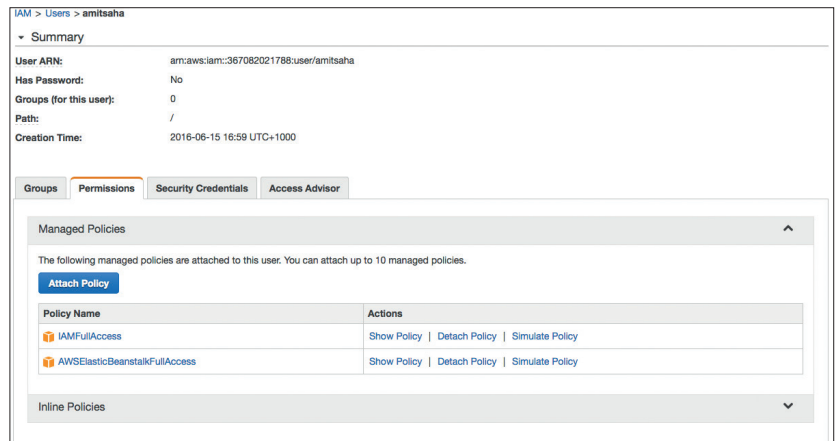


Figure 4: Policies attached to the user.

```
func main() {
    http.HandleFunc("/", handleRequest)
    http.ListenAndServe(":5000", nil)
}
```

We import the **runtime** package from the standard library, which we then use to get the Go version using the **Version()** function. Now, when a client connects to our application, we will respond with the Go version in addition to the greeting. Let's build an image from our Dockerfile next:

```
$ sudo docker build -t amitsaha/linux_voice_5 .
```

..

The **-t** option is used to specify an image tag and is usually of the form **username/image_name**. Once the image has been built, we will start a container with it:

```
$ sudo docker run -P amitsaha/linux_voice_5
```

From another terminal, if you do **sudo docker ps**, you should see a number of columns in the output; the **PORTS** column tells us that we have the port 32768 listening on our host, which maps to the port 5000 in the container on which our web application is listening on. Hence, **\$ curl 127.0.0.1:32768** will return us:

```
$ curl 127.0.0.1:32768
```

```
Hello World. Running: go1.6.2
```

This proves that our Dockerfile works. Let's stop the running container using **\$ sudo docker stop <your-container-name>**, where the name from the **docker ps** command. Now, let's add everything to Git and commit:

```
$ git add -A .
```

```
$ git commit -m "Dockerized version"
```

We are now ready to deploy. However, since we will now use a different platform, we will first terminate our current environment using **eb terminate**. Once the termination is complete, let's select the Docker 1.9.1 platform:

```
$ eb platform select
```

```
It appears you are using Docker. Is this correct?
```

```
(y/n): y
```

```
Select a platform version.
```

```
1) Docker 1.9.1
```

```
2) Docker 1.7.1
```

```
3) Docker 1.6.2
```

```
(default is 1): 1
```

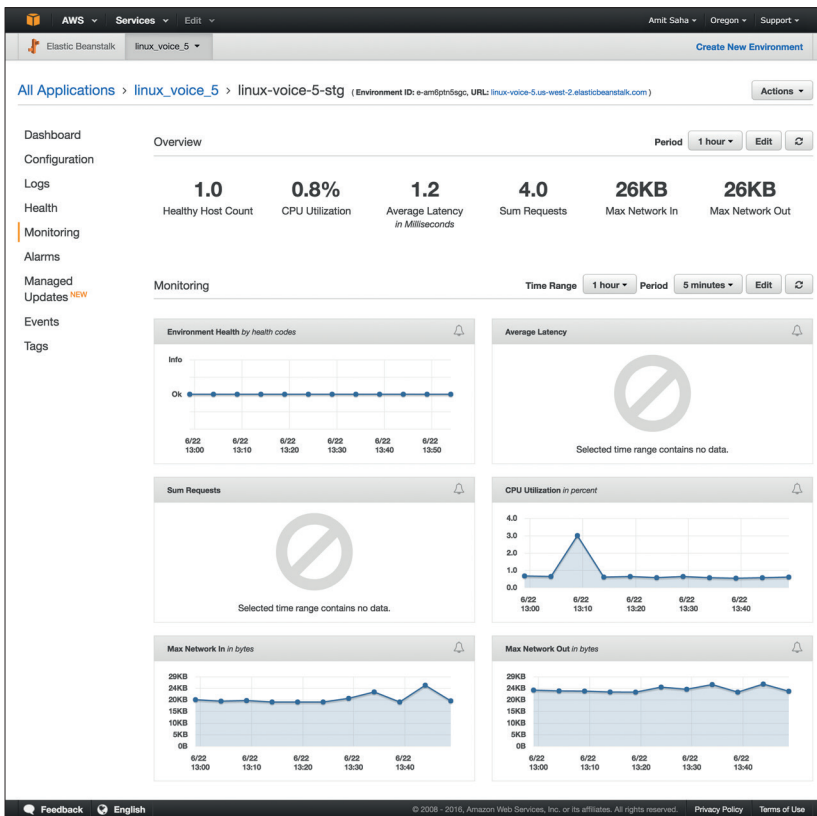


Figure 5: Monitoring dashboard.

Our platform has now been changed successfully, so we will now create a new environment with **eb create**. Once the environment creation has been completed, if we visit <http://linux-voice-5-dev.us-west-2.elasticbeanstalk.com>, we should get the response as "Hello World. Running: go1.6.2". This tells us that our web application is now up and running in the docker container we created with our Dockerfile. Next, we will modify our web application to do something slightly more useful.

Integer obfuscation as a service

Hashids (<http://hashids.org>) is a way of obfuscating integers (possible database identifiers) to a randomly generated string. It can be useful in any scenario where you don't want to expose the database identifier to your user. We will now update our web application's source code to implement a Hash ID as a service. When we pass in an integer to the service, we will get a random string back. The request will be of the form <http://linux-voice-5.us-west-2.elasticbeanstalk.com/?id=1> and we will get a JSON response back of the form `{"id":1,"obfuscated_id":"BARw3W"}`.

You can see the entire code at https://github.com/amitsaha/linux_voice_5/blob/master/application.go. Here I will discuss the main changes from our last version. At the beginning of the file we have a bunch of additional imports. The package we use to generate obfuscated strings is the third-party package github.com/speps/go-hashids. In addition, we also import two other standard library packages `math/rand` and `strconv`.

The function `getHashId()`, which performs the obfuscation, is as follows:

```
func getHashId(id int) (string, error) {
    hd := hashids.NewData()
    hd.Salt = strconv.Itoa(rand.Int())
    hd.MinLength = 6
    h := hashids.NewWithData(hd)
    return h.Encode([]int{id})
}
```

`func getHashId(id int) (string, error)` defines a function `getHashId()` that takes in an integer parameter `id` and returns two values – one of type `string` and the other of type `error`. In the first three statements we set up our `hashids` generator. The `salt` value should be a string and is set to a random integer converted to a string. Since the salt will be set to a random string everytime we generate a hashid, the obfuscated string will be different even when called with the same integer. We also set the minimum length of the obfuscated string to 6. This ensures that the obfuscated string is at least six characters in length. The last two statements generate the obfuscated string and return it.

The `handleRequest()` function is changed to the following:

```
func handleRequest(w http.ResponseWriter, r *http.Request) {
    id, err := strconv.Atoi(r.URL.Query().Get("id"))
    if err != nil {
        http.Error(w, "Bad id supplied", 400)
    } else {
        generatedId, err := getHashId(id)
        if err != nil {
            http.Error(w, "Error generating Id", 500)
        } else {
            fmt.Fprintf(w, generatedId)
        }
    }
}
```

The first statement in this function extracts the value passed via the `id` query parameter using `r.URL.Query().Get("id")` and converts it to an integer using the function `Atoi()` from the `strconv` package. If a non-integral value was specified for `id` or it was not specified, the value of `err` is not nil and we return 400 HTTP error with the message "Bad id supplied". If the passed-in value of `id` could be successfully converted, we next call the `getHashId()` function with the converted value. If there was some problem in generating the obfuscated string, we send an error with a 500 HTTP response. Else, we write the `generatedId` to the `ResponseWriter` object, `w`, using the `Fprintf()` function.

Overwrite your current `application.go` with the one at https://github.com/amitsaha/linux_voice_5/tree/ded80e07cd90278ba3501201632faa02de886157. Before we can run our application and try it out, we will fetch the `go-hashids` package by executing the following command while being in the `$HOME/work/golang/src/github.com/amitsaha/linux_voice_5`

directory:

```
$ go get .
```

```
..
```

Now, we can start our web application as earlier:

```
$ go run application.go
```

From another terminal, let's try sending in a couple of requests:

```
$ curl 127.0.0.1:5000?id=1
```

```
dvNDnp
```

```
$ curl 127.0.0.1:5000?id=abd
```

```
Bad id supplied
```

Our web application is now ready to be deployed. But first, we will need to update our Dockerfile to the following:

```
FROM golang
```

```
ADD application.go /go/src/github.com/amitsaha/linux_voice_5/application.go
```

```
RUN cd /go/src/github.com/amitsaha/linux_voice_5/ && go get -d -v .
```

```
EXPOSE 5000
```

```
CMD ["go", "run", "/go/src/github.com/amitsaha/linux_voice_5/application.go"]
```

The additional command `RUN cd /go/src/github.com/amitsaha/linux_voice_5/ && go get -d -v .` is needed for fetching the `go-hashids` package as we did earlier.

Deploying the updated application

Our web application is now updated and we have verified that it works as we expect it to. We have also updated the Dockerfile to reflect the changes. Let's stage the changes to Git and create a new commit with our changes:

```
$ git add -A .
```

```
$ git commit -m "Obfuscation service"
```

Now, to deploy our updated application, we will use the `eb deploy` command:

```
$ eb deploy
```

```
Creating application version archive "app-ded8-160617_094945".
```

```
Uploading linux_voice_5/app-ded8-160617_094945.zip to S3. This may take a while.
```

```
Upload Complete.
```

```
INFO: Environment update is starting.
```

```
INFO: Deploying new version to instance(s).
```

```
INFO: Environment health has transitioned from Ok to Info. Application update in progress on 1 instance. 0 out of 1 instance completed (running for 42 seconds).
```

```
INFO: Successfully built aws_beanstalk/staging-app
```

```
INFO: Docker container d4ee18f8cbe9 is running aws_beanstalk/current-app.
```

```
INFO: New application version was deployed to running EC2 instances.
```

```
INFO: Environment update completed successfully.
```

If you now go to the URL `http://linux-voice-5-dev.us-west-2.elasticbeanstalk.com/?id=1` in your browser or using the `curl` command, you will see that a string is returned.

Setting up a production environment

We now have our web application running at `http://linux-voice-5-dev.us-west-2.elasticbeanstalk.com`; we used the CNAME `linux-voice-5-dev` to indicate that this is going to be our development instance of the web application. We are convinced that we are happy with the state of the application now, or in other words production-ready. Now, we will set up our production environment:

```
$ eb create
```

```
Enter Environment Name
```

```
(default is linux-voice-5-dev): linux-voice-5
```

```
Enter DNS CNAME prefix
```

```
(default is linux-voice-5): linux-voice-5
```

```
...
```

```
..
```

This new environment is called `linux-voice-5` and the web application will be available at `linux-voice-5.us-west-2.elasticbeanstalk.com`. Once the environment has been successfully created, try making a few requests to it and make sure our web application is behaving as expected.

At this stage, we have two environments for our application – one for dev and the other production. We can list our current environments using `eb list`:

```
$ eb list
```

```
linux-voice-5
```

```
* linux-voice-5-dev
```

The environment with a `*` indicates the current

The screenshot shows the AWS Management Console interface for Autoscaling Policies. The 'Scaling Policies' tab is selected, showing a list of policies. Two policies are visible:

- ScaleDownPolicy:**
 - Execute policy when:** awseb-e-am6ptn5sgc-stack-AWSEBCloudwatchAlarmLow-AGR7XZ8Y1GCV breaches the alarm threshold: NetworkOut < 2000000 for 300 seconds for the metric dimensions AutoScalingGroupName = awseb-e-am6ptn5sgc-stack-AWSEBAutoScalingGroup-11MAWIDNJBUL6
 - Take the action:** Remove 1 instances
 - And then wait:** 360 seconds before allowing another scaling activity
- ScaleUpPolicy:**
 - Execute policy when:** awseb-e-am6ptn5sgc-stack-AWSEBCloudwatchAlarmHigh-1UI5F81LSI1 breaches the alarm threshold: NetworkOut > 6000000 for 300 seconds for the metric dimensions AutoScalingGroupName = awseb-e-am6ptn5sgc-stack-AWSEBAutoScalingGroup-11MAWIDNJBUL6
 - Take the action:** Add 1 instances
 - And then wait:** 360 seconds before allowing another scaling activity

Figure 6: Autoscaling Policies.

default environment. This means that if you do not specify an environment name to a command that accepts an environment name, this is the environment against which the operation will be performed.

Monitoring your application instances

We now have the first version of our web application up and running in two environments, and we would like to keep an eye on how the instances on which our applications are running are doing. We can do so using the **eb health** command. It will show us the platform we are running, the overall health of the environment, instance specific requests/second, percentage of requests that were served with different HTTP status codes and others. For an interactive view, we can use the **eb health --refresh** command.

The command also takes an environment name as the parameter, hence **eb health linux-voice-5** will show the health for the **linux-voice-5** environment. In addition, the Web UI for Elastic Beanstalk (<https://us-west-2.console.aws.amazon.com/elasticbeanstalk>) has a "Monitoring" tab for each environment (Fig 5).

Blue-green deployment

At this stage, our web application is running in production, and let's say that people are already depending on our service. Chances are that before long we would want to deploy a new version of our web application. Let's change our web application to now return the obfuscated number as a JSON response rather than in plain text. The repository at https://github.com/amitsaha/linux_voice_5 has the modified source code for **application.go**. I present the changes from our previous version below.

The first change is to import the **encoding/json** package from the standard library. The next change is to define a struct, **Response**:

```
type Response struct {
    Id      int `json:"id"`
    ObfuscatedId string `json:"obfuscated_id"`
}
```

We define a struct **Response** with two fields: **Id** of type **int** and **ObfuscatedId** of type **string**. For each we add a tag string in backticks, which tells the compiler that when we encode this structure as a JSON object, we want the field to appear as **id** and **obfuscated_id** respectively. If we don't do so the JSON object will have the fields as **Id** and **ObfuscatedId**, which is not usually the convention for JSON objects.

```
r := Response{Id: id, ObfuscatedId: generatedId}
w.Header().Set("Content-Type", "application/json")
json.NewEncoder(w).Encode(r)
```

In the first line of the above block, we create a variable of type **Response** object, **r**. Next, we set the "Content-type" header to **application/json**. Finally, in the last line we create a new JSON encoder that writes to **w**, and write the response variable encoded as JSON. Once the changes are committed to our repository, we're ready to deploy the update. First, we

will deploy to the development environment, **linux-voice-5-dev**:

```
$ eb deploy linux-voice-5-dev
```

```
...
```

Using **eb deploy linux-voice-5-dev** we deploy the current version of our application to the **linux-voice-5-dev** environment. Once the deployment has completed, we can make a request to see if our change is now live:

```
$ curl http://linux-voice-5-dev.us-west-2.
elasticbeanstalk.com/?id=1
{"id":1,"obfuscated_id":"ePgm6r"}
```

OK, so things are working as expected in the development environment. It's worth noting that this update happens in place on the instances. This means for a window of time, your web application won't be receiving any requests when the update is happening. In production, we don't want this to happen, or in other words we want to aim for a zero-downtime deployment. We can achieve this using the following approach:

- 1 Create a new environment with the updated version of our application (we will call the environment **linux-voice-5-stg**).
- 2 Verify it works as expected.
- 3 Swap the DNS from the current production environment to now point to the environment we created in step 1.
- 4 Terminate the old production environment.

This is usually referred to as "blue green" deployment. First, we create the new environment **linux-voice-5-stg**:

```
$ eb create linux-voice-5-stg
```

```
..
```

Once the environment is ready, we can find the CNAME of this environment as follows:

```
$ eb status linux-voice-5-stg
```

```
Environment details for: linux-voice-5-stg
```

```
Application name: linux_voice_5
```

```
Region: us-west-2
```

```
Deployed Version: app-7113-160620_231417
```

```
Environment ID: e-am6ptn5sgc
```

```
Platform: 64bit Amazon Linux 2016.03 v2.1.0 running
```

```
Docker 1.9.1
```

```
Tier: WebServer-Standard
```

```
CNAME: linux-voice-5-stg.f3tvjjma9v.us-west-2.
```

```
elasticbeanstalk.com
```

```
Updated: 2016-06-20 13:19:30.832000+00:00
```

```
Status: Ready
```

```
Health: Green
```

The CNAME of this environment has been randomly assigned, and we can make sure our web application is behaving as we expect it to by making a request such as **linux-voice-5-stg.f3tvjjma9v.us-west-2.elasticbeanstalk.com?id=1**. This completes Step 2 from above.

Now, we want to switch our current production environment (**linux-voice-5**) to switch to this one:

```
$ eb swap linux-voice-5-stg -n linux-voice-5
```

```
INFO: swapEnvironmentCNAMEs is starting.
```

INFO: Swapping CNAMEs for environments 'linux-voice-5-stg' and 'linux-voice-5'.

INFO: 'linux-voice-5.us-west-2.elasticbeanstalk.com' now points to 'awseb-e-a-AWSEBLoa-EI8FCU3NGSQX-1272685286.us-west-2.elb.amazonaws.com'.

INFO: Completed swapping CNAMEs for environments 'linux-voice-5-stg' and 'linux-voice-5'.

```
$ eb status linux-voice-5-stg
```

Environment details for: linux-voice-5-stg

Application name: linux_voice_5

Region: us-west-2

Deployed Version: app-7113-160620_231417

Environment ID: e-am6ptn5sgc

Platform: 64bit Amazon Linux 2016.03 v2.1.0 running Docker 1.9.1

Tier: WebServer-Standard

CNAME: linux-voice-5.us-west-2.elasticbeanstalk.com

Updated: 2016-06-21 00:32:03.618000+00:00

Status: Ready

Health: Green

Unfortunately, AWS Elastic Beanstalk doesn't currently have a feature to rename environments, which means that we will need to mentally remember which is currently our production environment using some convention. Our production web application is now running in the **linux-voice-5-stg** environment, so we can safely terminate our old production environment now:

```
$ eb terminate linux-voice-5
```

Scaling the number of application instances

We have an auto scaling group set up for our application in each environment, which enables us to scale up or scale down the number of instances. An auto scaling group specifies the minimum number of instances, maximum number of instances and the desired number of instances we want running at any given point of time. A scale up operation happens when the current number of instances is less than the maximum number of instances and the specified metric dimension exceeds the metric threshold. Conversely, a scale down operation happens when the number of instances running is more than the desired number of instances and the specified metric dimension is less than the metric threshold. This will be clearer when we consider the auto scaling group that Elastic Beanstalk automatically creates for us.

When we go to <https://us-west-2.console.aws.amazon.com/ec2/autoscaling/home> (assuming us-west-2 is your AWS region) via your browser, you will see two auto scaling groups, one for each of the two environments we currently have deployed our application to. When you click on any of the scaling group, you will see the two policies defined, one for scaling up and the other for scaling down (Figure 5). The scaling down policy for the group states the execution policy as "NetworkOut < 2000000 for 300 seconds" and the scaling up policy states "NetworkOut > 6000000 for 300 seconds". The

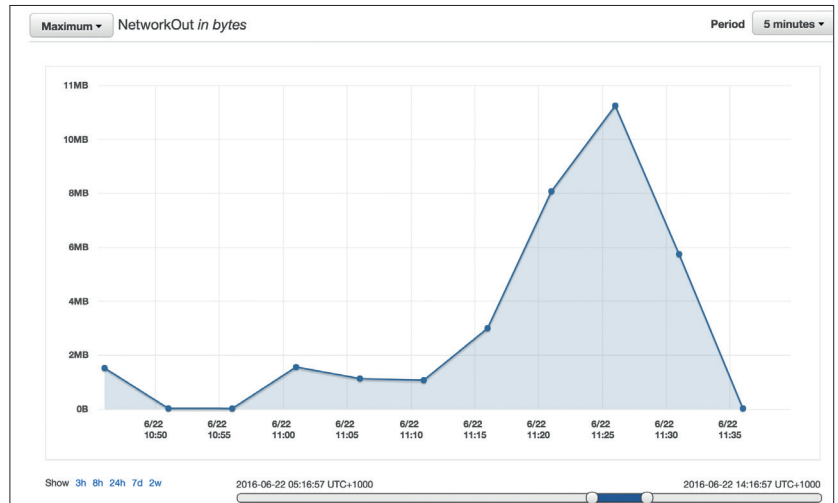


Figure 7: Maximum NetworkOut for our instance during a high-traffic window

duration of 300 seconds is referred to as the breach duration. Hence, the scale up policy here is triggered if the the average total number of bytes exceeds 6000000 bytes (6MB) for more than 300 seconds and alternatively the scale down policy is triggered if the average total number of out bytes is less than 2000000 (2MB) for more than 300 seconds.

We will use the **wrk** (<https://github.com/wg/wrk>) tool to simulate traffic to our web application. After a bit of trial and error, I found that the following would result in 6MB NetworkOut bytes for over five minutes:

```
$ wrk -t50 -c500 -d5m http://linux-voice-5.us-west-2.elasticbeanstalk.com/?id=1
```

We can see from Figure 7 that in the time window between 11.19 and 11.35 the maximum **NetworkOut** exceeded 6MB for more than five minutes. This triggered the scale up operation and a new instance will be added to your environment. Once your traffic again reduces to less than 2MB for more than five minutes the older instance in the environment will be automatically terminated.

In most cases, a web application would be backed by a database. Amazon Relational Database Service (RDS), also part of AWS, allows a choice between *MySQL* and *PostgreSQL* (among others). An RDS instance can be created as part of your Elastic Beanstalk application environment using **eb create --database**. The drawback to this approach is that if you terminate your environment as part of a deployment as we saw in blue-green deployment, it brings down the RDS instance associated with it too. Hence, it is recommended that you manage your RDS instance separately. Once we have our RDS instance set up, we will use an appropriate driver listed at <https://github.com/golang/go/wiki/SQLDrivers> to interface with the database server.

The GitHub repository at https://github.com/amitsaha/linux_voice_5 has the final web application code along with resources to learn more about Elastic Beanstalk, Docker and Golang. 📖

Amit Saha is the author of *Doing Math with Python* (No Starch Press) and a software engineer. He blogs at <https://echorand.me>, tweets @echorand and can be reached via email at amitsaha.in@gmail.com.



Valentine Sinitsyn develops high-loaded services and teaches students completely unrelated subjects. He also has a KDE developer account that he's never really used.

CORE TECHNOLOGY

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

Extended attributes & POSIX ACL

In Linux, pretty much everything is a file. This includes your documents and executable programs, directories (or folders), devices and even IPC objects such as Unix sockets. Each file has a name and a size, along with some timestamps, access permissions, and other associated metadata.

Nine times out of ten, this is everything you need. But imagine you want your own, custom bit of metadata. This could be a tag or a URL you've downloaded the file from. It could also be some permission: "Alice and Bob can read this". Linux makes this possible with extended attributes, or simply "xattrs".

Extending file metadata

In a nutshell, extended attributes are key-value pairs associated with a filesystem object. Attribute names (or keys) are strings. Values can be anything: at this point, they are just sequences of bytes. Moreover, keys are namespaced, and attribute names are always **namespace.something**. Namespaces are also called attribute classes. The kernel currently recognises four of them (see **xattr(7)**).

security.* is for kernel security modules. For example, SELinux stores file context here. File capabilities also rely on this mechanism. Then, there is a **system.*** namespace. POSIX Access Control Lists (ACL) live there, as we'll learn in a moment. Trusted extended attributes live in **trusted.*** are available only to trusted userspace processes. A process needs

to have **CAP_SYS_ADMIN** capability to read or write trusted extended attributes. This usually means it must run as root. The GlusterFS distributed filesystem implements many of its features via trusted extended attributes.

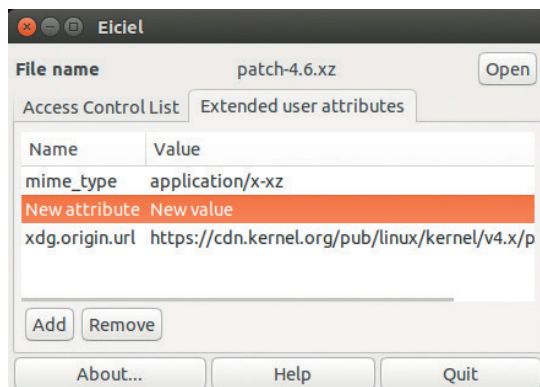
Finally, there is extended user attributes and **user.*** namespace. This is where you put the metadata you want. Of course, this is also subject to permission checks; you can't write extended attributes if you can't write to a filesystem object, and you can't read or list them unless you have the read permission. Moreover, only regular files and directories may carry extended user attributes. They are commonly employed to store the file's original URL, MIME type, or character set. **Freedesktop.org** defines some attributes (<https://www.freedesktop.org/wiki/CommonExtendedAttributes>) for conforming applications to use. However, nothing prevents you from creating your own ones; just think of a sufficiently unique name to prevent clashes. A popular recommendation is to prefix your custom attribute with the reversed domain name you own.

Extended attributes naturally need some support at the filesystem's side. Linux has many filesystems, and the good news is that all major players in this field provide xattrs. This includes both ext4 and btrfs, which you likely have on your PC or server. Distributed filesystems, say HDFS or the aforementioned GlusterFS, implement them as well.

There are a couple of commands you may use to work with extended attributes from the shell. In Ubuntu, they come with the **attr** package, which you are not likely to have by default. So, install it via **apt-get**. In other major distributions, the package name is the same, or similar.

To get extended attributes from a file, you use the **getfattr** command. It understands a few options: **-d** dumps all extended attributes that match the filter that **-m** sets. This defaults to **^user**, or extended user attributes only. For anything else, you'd want to adjust this regular expression accordingly. A typical file doesn't have too many extended attributes, so you may disable the filter altogether with **-m -**.

Eiciel (<https://rofi.roger-ferrer.org/eiciel>) is a graphical extended attributes/ACL editor for Gnome. It also appears as a tab in *Nautilus*.



If you already know the name of the attribute you are looking for, tell **getfattr** with **-n**. As extended attributes are binary values in general, the command encodes them when printing. By default, Base64 is used, but you can force hexadecimal or text encoding with **-e**. **-R** makes **getfattr** recursive. This way, you can list extended attributes on everything below the given directory. This is helpful sometimes, as **find** knows nothing about extended attributes.

In the wild

By now, you may start wondering where you can find extended attributes in your system. Recall that some programs already use them for various purposes. Everything we need is a list of candidates to hunt for.

Personally, I'm a *Netscape*, then *Mozilla*, then *Firefox* user since the late 90s. The rest of the world seems to opt for Google *Chrome* (or at least *Chromium*) in 2016. If you are from this camp, try this:

```
$ cd ~/Downloads
$ getfattr -d getfattr -d patch-4.6.xz
# file: patch-4.6.xz
user.xdg.origin.url="https://cdn.kernel.org/pub/linux/kernel/v4.x/patch-4.6.xz"
user.xdg.referrer.url="https://www.kernel.org/"
```

Of course, you should supply **getfattr** something you downloaded recently instead of my **patch-4.6.xz**. You see that *Chromium* remembers the file's original URL and the page it was downloaded from, or the referrer. Isn't it handy, especially if you can't remember where you got this file? If you aren't a *Chrome* fan (hooray!), the **curl** command line tool can do much the same. You only need to send it an **--xattr** option:

```
$ curl -s -o patch-4.6.xz --xattr https://cdn.kernel.org/pub/linux/kernel/v4.x/patch-4.6.xz
$ getfattr patch-4.6.xz
# file: patch-4.6.xz
user.mime_type="application/x-xz"
user.xdg.origin.url="https://cdn.kernel.org/pub/linux/kernel/v4.x/patch-4.6.xz"
```

The attributes are slightly different now. **curl** doesn't know the page you get the link from, so it doesn't set the referrer. Still, they are "common extended attributes", in the FreeDesktop specification's sense.

Making your own

You can also create your own custom attributes with **setfattr** tool. It's simple:

```
$ setfattr -n user.tag -v todo somefile
$ getfattr -n user.tag somefile
# file: somefile
user.tag = "todo"
```

This way, you can implement file tags at the filesystem level. Below is a quick way to find all files marked with the **todo** tag in your home directory:

```
$ getfattr -R -n user.tag ~ | grep file:
```

If this yields more than a handful of items, perhaps you want to rethink your time management strategy.

You may also use extended attributes to store the file's hash sum, such as MD5 or SHA256:



```
$ setfattr -n user.sha256sum -v "$(sha256sum -b somefile)" somefile
```

```
$ getfattr -n user.sha256sum --only-values somefile | sha256sum -c somefile: OK
```

We need the double quotes with **`setfattr`** (LV027), as **`sha256sum`** output contains embedded spaces. Moreover, we supply the **--only-values** switch to dump the raw attribute value. This is a previous **sha256sum** output in this case. If we used trusted extended attributes instead of user ones, this could serve as a reliable integrity check method.

However, not every Linux command preserves extended attributes when copying or moving your files around. Let's do some quick experiments:

```
$ setfattr -n user.attr -v 1 somefile
$ cp somefile somefile_copy
$ mv somefile anotherfile
$ getfattr -n user.attr somefile_copy anotherfile
somefile_copy: user.attr: No such attribute
# file: anotherfile
user.attr="1"
```

You see that **mv** preserves extended attributes while **cp** doesn't – at least, that's the default behaviour. **tar** and **rsync** are like **cp** in this sense. This is because moving a file keeps its inode number the same, and extended attributes are really associated with inodes. Even if **mv** works across filesystem boundaries, it has to emulate this behaviour.

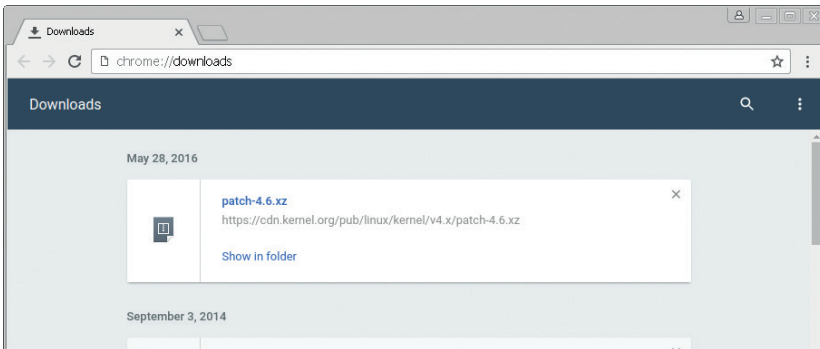
You may tell **cp** and friends to preserve extended attributes as well. For **cp**, **--preserve=xattr** does the trick. Other tools accept **--xattr (rsync)** or **--xattr (tar)**, akin to **curl**. The bottom line is you shouldn't treat extended attributes and the file as something indivisible. An unintended **cp** may rip everything off.

Recall that extended user attributes aren't the only namespace. Here's what I have attached to a humble **ping** on my Fedora 23 box:

```
$ getfattr -m - -d /usr/bin/ping
# file: usr/bin/ping
security.capability=0sAQAAAqAwAAAAAAAAAAAAAAAAAAAA=
security.selinux="system_u:object_r:ping_exec_t:s0"
```

The first attribute, **security.selinux**, defines the SELinux security context for the file. The second, **security.capability**, defines the file capabilities. We discussed them back in LV023, but in case you missed that issue, capabilities are what give **ping** the

FreeDesktop has some recommendation for extended user attributes. Consult them before reinventing the wheel.



Every file you download in Chrome gets a couple of extended attributes as a bonus. Well done, Google!

right to create raw network sockets. Otherwise, this is a privileged operation available to the **root** user only.

You surely know how traditional file permissions work in Linux. There are three groups of bits responsible for read, write and execute permissions for the owner, group and everyone else, respectively. There are also some special bits like the sticky one, but they don't change the overall picture.

Traditional Unix permissions do their job quite well. But there are some rare configurations that you can't express in their terms. Say, Alice wants read and write permissions for herself and Bob, read permission for the group and nothing for the rest of the world.

POSIX ACL comes to the rescue. If you have some experience administering Microsoft Windows, you already know the concepts. Each filesystem object has a set of associated users and groups, along with their respective access rights. As there is no hard division by owner, group and the world, ACL allows for much greater flexibility. This is hardly a feature you'll use often, yet it may come up useful someday.

More specifically, POSIX ACL is a set of entries consisting of a type, a qualifier, and a set of permissions (see **acl(5)**). There are six distinct entry types. Three of them are the traditional owner, group and others, known as **ACL_USER_OBJ**, **ACL_GROUP_OBJ** and **ACL_OTHER**. Then there's **ACL_MASK**, which plays a role similar to **umask**. That is, it contains maximum permissions for the file or directory. If some permission bit is unset in **ACL_MASK**, setting it for any user (other than the owner) or group will have no effect. Two other types, **ACL_USER** and **ACL_GROUP**, define access permissions for named users and groups.

A natural question now is how POSIX ACL and traditional Unix permissions interplay. The answer is they are always in sync. If you change file owner permissions, **ACL_USER_OBJ** entry is updated, and vice versa. Group permissions are little more tricky. They are mapped to **ACL_MASK**, if it is present. If not, they are mapped to **ACL_GROUP_OBJ**. World permissions are always synchronised with **ACL_OTHER**. In other words, POSIX ACL is an extension mechanism which complements traditional Unix file permissions. You can think in terms of POSIX ACL exclusively, yet it is rarely useful.

It is also natural for files within a single directory to share similar permissions. Otherwise, why do you put

them together? POSIX ACL facilitates this with the default ACL that you can assign to a directory. Later, when you create something within that directory, it will automatically get a copy of the default ACL to start with. The directory's default ACL doesn't have to coincide with its own ACL. Say, you may have a directory executable for Bob, so he can list its contents. This doesn't automatically mean that Bob will be able to execute any file within the directory. Files in the directory may also have an ACL other than the default. It's adjustable with the commands we'll see in a moment.

Finally, you want some notation to communicate POSIX ACL to the system. There are two of them: the long text form and the short text form. In either case, the entry is represented as a colon-separated string. Its fields are the type, the qualifier (a user or group name or ID), and permission bits. The latter use well-known **rwX** notation. In long text form, each entry comes on a separate line. The hash mark (**#**) starts a comment. If there is an **ACL_MASK** entry present, a comment describes effective access rights. Consider this:

```
user::rwX # ACL_USER_OBJ entry
user:alice:rwX # effective: rw-
group::rw- # ACL_GROUP_OBJ entry
mask::rw-
other::r--
```

Here, the owner has full permissions. Alice wants the same, but **ACL_MASK** disables the **x** bit for her. The group can read and write. If it also wanted to execute, **ACL_MASK** would disallow it. Everyone else can read, and that's all.

The short text form is more compact. Entries are comma-separated, and no comments are allowed. You may also abbreviate types as **u**, **g**, **m** and **o**. The long text form is intended to be human-readable; short text form is mostly an interchange format.

Time to play

Now, let us play with POSIX ACL a bit. To manipulate access rights, you'll need several command-line tools which come with the **acl** package. You should be able to find one in your distribution's repositories. The **getfacl** tool displays ACL for the filesystem object:

```
$ getfacl somefile
# file: path/to/somefile
```

Do it in Python

This month, we look at extended attributes and POSIX ACL from the shell angle. However, the commands we covered are by no means the only interface to these features.

For more complex scripting needs, consider two Python bindings: **xattr** (<https://pypi.python.org/pypi/xattr>) and **pyxattr** (<http://pyxattr.k1024.org>). The former claims to support more platforms: Linux and Mac OS X plus FreeBSD and Solaris in experimental status. The latter has a somewhat cleaner API and is better documented. **xattr** also provides a command-line tool of the same name. You can use it to manage extended attributes much the same way you do with **getfattr** and **setfattr**.


```
# owner: val
# group: val
user:rw-
group:r--
other:r--
```

You see it uses the long text form. **-t** switches to tabular format, and you can turn off comments with **-c**. **-d** dumps the default ACL for an object. You can say **somefile** doesn't have any POSIX ACL assigned – otherwise, there would be more than three entries. Let's fix this:

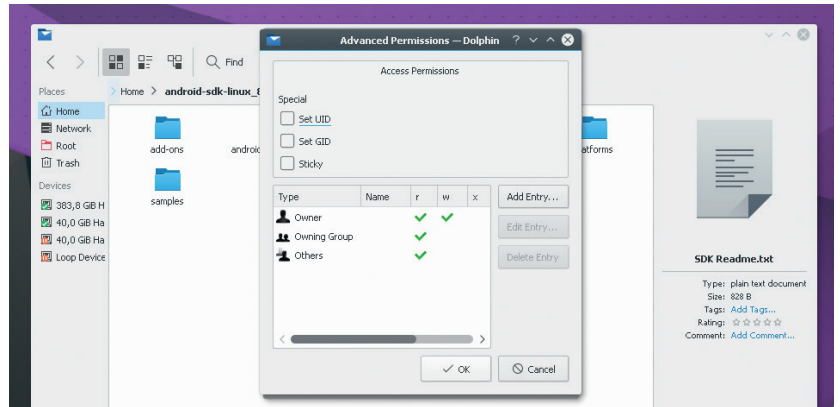
```
$ setfacl -m user:alice:rw somefile
$ getfacl -c somefile
user:rw-
user:alice:rw-
...
```

We granted Alice a write permission to **somefile** with **setfacl -m** tells us that we want to modify the existing access list. If we wanted to revoke a permission, we'd use **-x**. It is also possible to create a default ACL for a directory with **setfacl -d**:

```
$ mkdir someplace
$ setfacl -d -m user:alice:rw someplace
$ touch someplace/somefile
$ getfacl -c someplace/somefile
user:rw-
user:alice:rw-
...
```

Instead of providing permission entries at the command line, you can use the **-M** and **-X** options. They accept the names of files containing the ACL you want to apply. You can easily tell when an object has some POSIX ACL entries. **ls -l** marks those files or directories with a plus sign:

```
$ ls -l
-rw-rw-r--+ 1 val val 0 Jun 7 00:29 somefile
drwxrwxr-x+ 2 val val 4096 Jun 7 00:40 someplace
```



POSIX ACL are really extended attributes that the kernel recognises and handles appropriately. Our good old friend **getfattr** proves it easily:

```
$ getfattr -m - -d somefile
# file: somefile
system.posix_acl_access=0sAgAAAAEABgD///// ...
$ cp somefile anotherfile
$ ls -l anotherfile
-rw-rw-r-- 1 val val 0 Jun 7 00:53 anotherfile
```

Note there is no **+**. Again, **mv** preserves POSIX ACL, unless the target filesystem doesn't provide support for extended attributes.

Extended attributes and POSIX ACL are quite simple yet are somewhat a disguised feature. Their unpopularity is not a consequence of some design flaw; extended attributes are a low-level mechanism, so you don't always have to interface with it directly. POSIX ACL is a bit too flexible for a general Linux system. It is fun to play with, but if you use POSIX ACL in production, please drop us a note.

KDE (well, the *Dolphin* file manager) understands POSIX ACL and lets you manage them out of the box.

Command of the month: **chattr** and **lsattr**

Linux recognises quite a few attributes, but only a handful of them are seen in the wild. Many attributes are related to dynamic file compression. The data is stored "zipped" on the disk and uncompressed on the fly when you read it. There is an attribute to wipe deleted files with zeros. And there is also an attribute that makes file undeletion possible.

Traditional file attributes are case-sensitive single-letter values. For instance, **A** prescribes not to update the file's access time. This may save some power on disk I/O, especially on laptops with magnetic hard disks. **i** makes a file immutable. No user, including root, can move or delete an immutable file, or create a hard link to it. Only a superuser or somebody with **CAP_LINUX_IMMUTABLE** capability can add this attribute to a file or remove it once added.

The **e** attribute is read-only, and it is a marker that the file is using filesystem extents. You'll find the list of

all supported attributes in **chattr(1)**. Two commands exists to manage these attributes. First, there's **chattr** to set and unset them. The syntax is simple. You supply a single-letter attribute name prefixed with **+** to enable the attribute, or **-** to disable it. **=** causes **chattr** to overwrite the current attributes with those you specified. Say, **chattr +i somefile** makes **somefile** immutable. The **lsattr** command lists attributes for the files you specify, or everything in the current directory:

```
$ lsattr
-----e-- ./somefile
----i-----e-- ./immutable
```

Here, both files are using filesystem extents, and **immutable** is, well, immutable:

```
$ sudo rm immutable
rm: cannot remove 'immutable': Operation not permitted
Even the almighty root isn't that mighty at times. 🚫
```

LINUX INSIDE: **THE FALCON 9**

Free Software at the cutting edge of space exploration

SpaceX's Falcon 9 rocket – shown here launching a Dragon spacecraft on a resupply mission to the International Space Station – uses a combination of liquid oxygen and kerosene to propel cargoes into space. Unlike other commercial rockets, the Falcon 9 is designed to be reusable, and on 8 April 2016, it became the first rocket to successfully land at sea when it touched down on a robotic drone-ship in the Atlantic. This reuse should enable SpaceX to dramatically lower the cost of space exploration, and it's all powered by the Linux kernel.



LINUXVOICE

This is what we've done in the last 24 issues.
Subscribe to the next 12 from just **£38**.



Every subscription includes access to every PDF, ePub and audio edition we've ever published.

shop.linuxvoice.com

