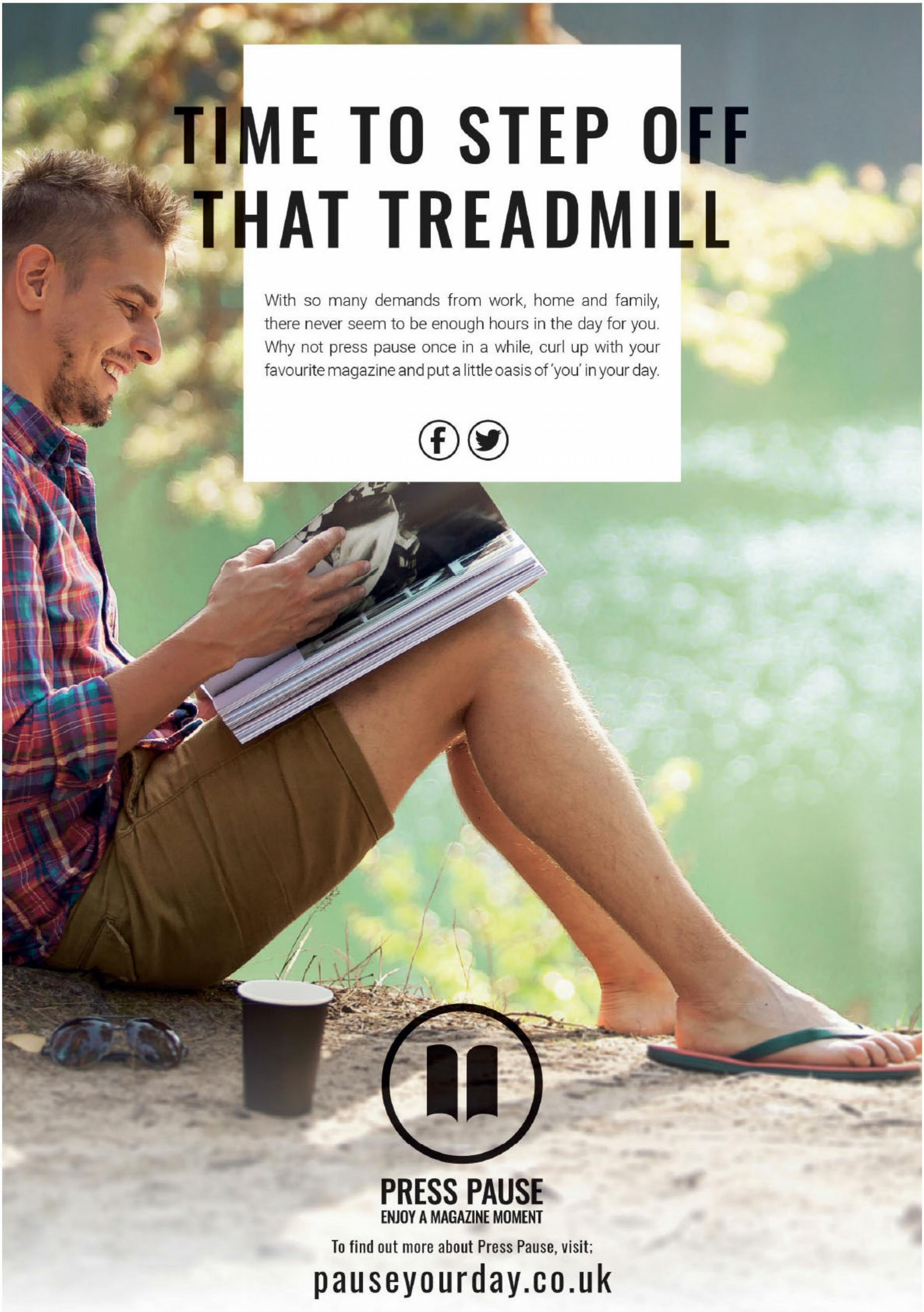

TIME TO STEP OFF
THAT TREADMILL
With so many demands from work, home and family, there never seem to be enough hours in the day for you. Why not press pause once in a while, curl up with your favourite magazine and put a little oasis of 'you' in your day.
PRESS PAUSE
ENJOY A MAGAZINE MOMENT
To find out more about Press Pause, visit;
pauseyourday.co.uk

GET STARTED
Boot Linux now with our Pop!_OS guide

EBOOK TIPS
Build a digital library with books new and old

FREE 4.7GB DVD
RUN LINUX TODAY!

LINUX FORMAT

The #1 open source mag

PLUS!
3 Pi PROJECTS
- ANALOGUE I/O
- LIVE STREAMING
- CLIMATE MONITOR

SORT OUT YOUR STORAGE

✔ Protect files ✔ Boost speeds
✔ Add RAID ✔ Monitor drives

65 pages of tutorials & features

Discover the best lightweight Linux desktops

Create better animated GIFs and videos

The Linux Foundation's
John Mertic
on why mainframe is as relevant today as ever

STREAM SERVER
Code a message client with Kafka

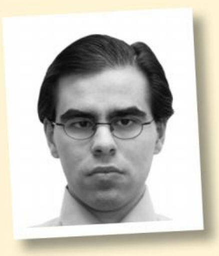
HACK MINECRAFT
Build your own interface for the classic game

FUTURE

WHO WE ARE

This issue Jonni is securing our stuff, so what's your worst data-loss experience?

Jonni Bidwell
Well, the dog ate my homework on more than one occasion. I'd use this opportunity to talk about the weird Btrfs errors that forced me to reformat two large hard drives, but credit to Btrfs – no data was ever lost, just forced into read-only mode within five minutes of being written to.



Tam Hanna
My worst ever catastrophe was data being transmitted to a new NAS, which then suddenly died during configuration and destroyed the filesystem of the hard disks. Fortunately, I could recover most of it – but it still was a very painful experience.

Nick Peers
My worst data-loss experience is my first one: 20 years ago, when building a new PC for someone else. Long story short, we got the data back, but via a command-line tool that required me to manually copy text files individually to floppy disk. There were over a thousand. That was a 'fun' Christmas.

Mayank Sharma
The worst was when I pointed *dd* to write an ISO to the backup drive. I realised my mistake and interrupted it midway, but even after weeks of combing the remains with *testdisk* all I got was about 10 per cent of the data back and a costly but valuable lesson.

Alexander Tolstoy
I once accidentally formatted the SD card full of my vacation photos and videos. I had to use the marvelous *photorec* utility to bring the files back. As for the rest, I regularly do back-ups, so there were perhaps no humorous cases with my storage.

» Win a secure Nitrokey Storage 2

Send your thoughts to the *Linux Format* dungeon server at **linuxformat@futurenet.com** and secure your chance to win a 32GB Nitrokey! It's the complete open hardware-encrypted storage solution!
Learn more at **www.nitrokey.com**.

Room at the inn?

Storage, storage everywhere, but not a bit of it is safe. Where do you store your stuff? On your boot drive? On a magnetised platter spinning at 7,200 revolutions per second? Amongst the electrons in a solid state drive? On someone else's server in the sky? There's no one storage solution that's perfect for everyone.

Perhaps you need the fastest access possible. Perhaps redundancy is an absolute must. Perhaps you need terabytes upon terabytes. Perhaps flexible access from anywhere is key. This issue we're letting Jonni loose on the storage problem. He's going to explore the many options available, explain how to create redundancy and store files in the cloud for easy access. As flexible as cloud storage is – it's great for simple off-site backup – you can't beat keeping things locally and in your own jurisdiction.

With the price of SSD storage crashing below 10p per gigabyte and old-school HDD storage bouncing around 2 to 3 pence per gigabyte, storage is cheap right now. So it's an ideal time to build that mirror or RAID6 you've always promised yourself! Let Jonni guide you through the jungle of storage types, file systems, configurations and options to get the perfect solution.

If all your files are already securely tucked away, we've plenty of other features and tutorials to keep you busy this summer. We test the best lightweight desktops you might want to try, we build a better ebook library, there's more fractal fun to be had with open source, why not livestream your gaming or maker antics with a Raspberry Pi – plus benchmarking and our get-started guide to the amazing Pop!_OS from System76. It's another packed issue, so enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com

Subscribe & save!
On digital and print
– see p24

Contents

REVIEWS

MSI GeForce GTX 1650 Gaming X 4G . . 19
Nvidia's new entry-level GPU brings a modest boost in performance thinks **Jarred Walton** – who explains what's new, what's cool and if it's worth the money.

Robolinux 10.5 20
Mayank Sharma isn't too gullible, but has an uncanny tendency to fall for distros that then don't deliver on the claims they make.

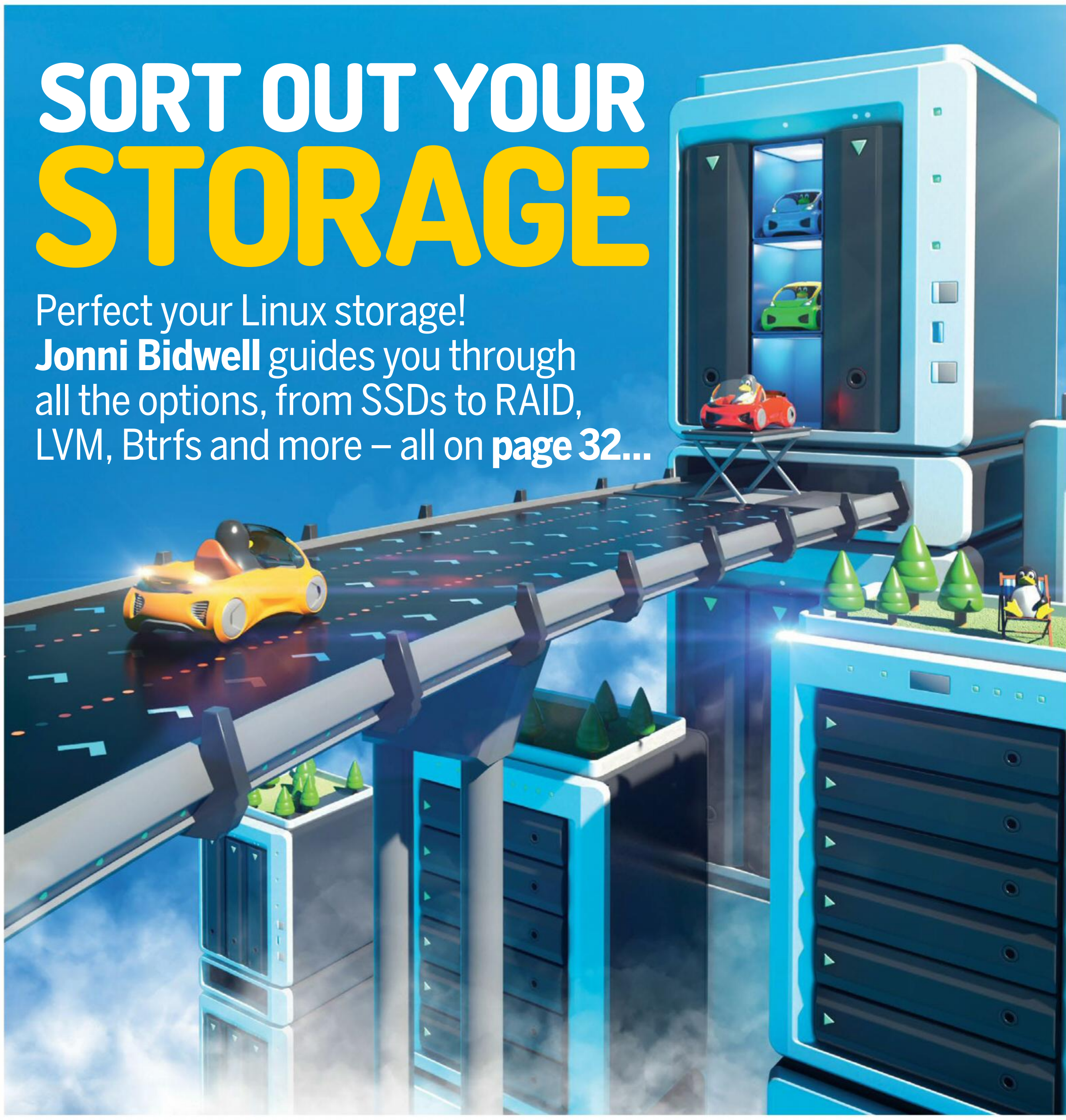
Emmabuntüs DE 2 1.04 21
Not a big fan of the kitchen-sink approach, **Mayank Sharma** finds one distro that's put in enough effort to perform well and has a heart of green behind it all...

Hexagon OS 22
A sucker for home-brewed apps, **Mayank Sharma** gets lured into a new distro that's headed in the right direction, but still has a long way to go.

OpenSUSE 15.1 23
This project has been in a state of unrest for some time now, and **Mayank Sharma** wonders if that uncertainty has left a mark on the distro.

SORT OUT YOUR STORAGE

Perfect your Linux storage! **Jonni Bidwell** guides you through all the options, from SSDs to RAID, LVM, Btrfs and more – all on **page 32...**



ROUNDUP

Lightweight desktops 26
With his gym membership going unused, **Shashank Sharma** decides to cut down the flab on his desktop instead by examining these faff-free desktops.

INTERVIEW

Picture in a mainframe 40
Jonni Bidwell wants to know if he could install Linux on the mainframe at *Linux Format* Towers. The Open Mainframe Project's **John Mertic** has all the answers.

RASPBERRY Pi USER

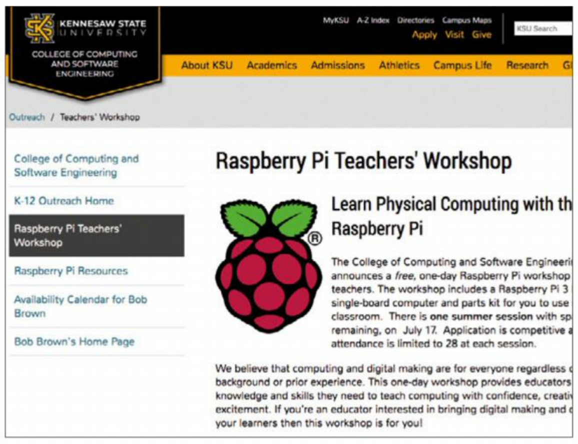
Raspberry Pi news 50
Raspberry Pi protecting worldwide wildlife, traffic modelling for autonomous cars, and Pi-based training programmes.

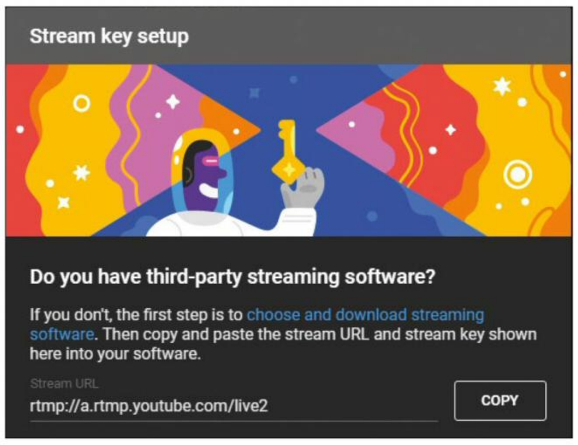
4tronix PiBug 2WD 51
Les Pounder takes a look at a simple robot that can be built in less than 30 minutes without the need for any special tools.

Using analogue HAT interfaces 52
Les Pounder shows us two more projects using the deceptively simple Explorer HAT Pro – and a little Python code.

Monitor humidity with the I2C bus .. 54
Tam Hanna feels moist and no one likes that, so he's monitoring his environment.

Live streaming 58
Christian Cawley explores how to create a YouTube livestream from your living room with a Raspberry Pi and camera module.



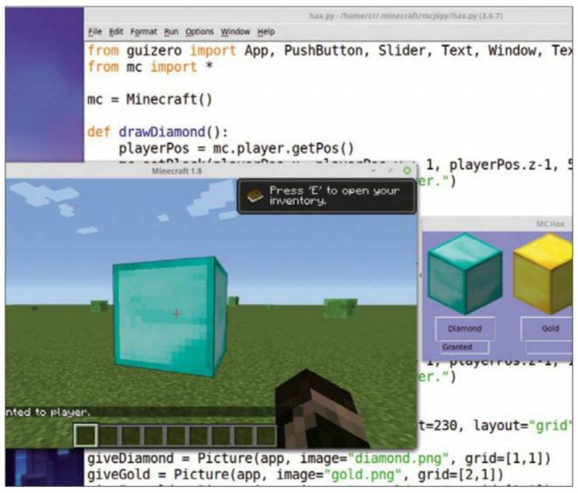


CODING ACADEMY

Parsing XML files 84
John Schwartzman shows how to parse and display XML files in Python and Go.

Apache Kafka tools 88
Mihalis Tsoukalos explains how to install and create utilities that use Apache Kafka.

Hacking Minecraft with Python 92
Calvin Robinson creates a custom toolbox for quickly manipulating a *Minecraft* environment, complete with GUI.



REGULARS AT A GLANCE

News 6
Google thinks you need adverts, South Korea thinks you want Linux, Unity now wants Linux, Nvidia plays nice, malware wants Linux, and RISC-V advances.

Linux user groups 11
Les Pounder goes all European (while he still can) at MiXiT 2019.

Answers 12
Making huge test files, clearing out your temp folder, changing the default mounting point, and monitoring logs.

Mailserver 16
What will IBM do with Red Hat? How to make Linux more accessible? What fonts can you use? And more Arch, please!

Subscriptions 24

Back issues 66

Overseas subs 67

HotPicks 78
Alexander Tolstoy doesn't have time to sit down and watch first-class HBO series *Chernobyl* – he's far too busy trying to get first-class open source to reach criticality like: *Hyper*, *DiffPDF*, *KWin-lowlatency*, *LanguageTool*, *Kepka*, *Password Safe*, *Devede NG*, *Stacer*, *WallGen*, *SuperTuxKart* and *OpenPatrician*.

Your free DVD 96

Next month 98

TUTORIALS

TERMINAL: McFly 60
While **Shashank Sharma** swears by the powers of *Bash*'s history expansion, there's another tool that can help you to be more efficient: *McFly*.

EBOOKS: Calibre 62
Nick Peers immerses himself in ebooks with a guide to viewing, managing, converting, editing and even streaming them from a server.

VIDEO: FFmpeg 68
Tired of stuttering GIFs on your blogs and websites? **Dan Dart** explains how to turn your best memes into smaller silent video files instead.

MATHS: More fun with fractals 70
Mike Bedford continues to review the curious world of fractals by delving into objects that mathematicians first described as "monsters".

BENCHMARKS: Sombrero 74
Ed Bennett shows how you can apply the same techniques used to benchmark the world's fastest supercomputers to one-up your home-desktop friends...

IN-DEPTH

Get started with... Pop!_OS 46
Discover the hip new kid on the distro block: Pop!_OS, from System76. Easy to use but super-powerful, testing it makes **Jonni Bidwell** drops his Arch addiction.

Newsdesk

THIS ISSUE: Google blocks ad-blockers (sort of) » South Korea hearts Linux » Unity Engine » HiddenWasp malware » BOOM

AD-FILLED INTERNET

Google ploughs ahead with ad-blocking restrictions

Google continues with its unpopular changes to ad-blocking extensions – unless you're an enterprise-class user. Typical!

Google, it appears, is planning to disable ad-blocking extensions in *Chrome* for everyone but enterprise users. In January 2019 Google first announced that it was planning to change *Chrome*'s extensions system, called Manifest V3, but many had hoped that the backlash to the proposed changes – which would dramatically impact the efficiency of ad-blocking extensions – would cause Google to change it mind. Unfortunately, it appears not.

As 9to5Google reports (see **http://bit.ly/LXF252ManV3**), Manifest V3 alters the permissions system for *Chrome*'s extensions, and stops the webRequest API in *Chrome* from being able to block ads and requests before they are downloaded. This is an essential component for many modern ad-blocking extensions, and the removal of that feature has worrying security implications. After all, if an ad does contain malicious code, you want to prevent it from downloading and showing on your device.

Critics of this move – of which there were plenty – were quick to point out these security concerns, prompting Google to issue a response (**http://bit.ly/LXF252GoogleResponse**): "*Chrome* is deprecating the blocking capabilities of the webRequest API in Manifest V3, not the entire webRequest API (though blocking will still be available to enterprise deployments)."

This response did little to allay people's concerns – in fact, it appears to have inflamed the situation. The contentious part of the company's response is the admission that ad-blocking will still be available to enterprise users of *Chrome*.

This means that enterprise users may still be able to use ad-blockers that rely on the webRequest API, or at least create their own extensions that block potentially malicious elements of webpages. However, this leaves regular *Chrome* users rather exposed.

It doesn't mean that ad-blocking extensions will disappear from *Chrome*, however. Instead, the extensions will now have to use a rules-based system called 'declaritiveNetRequest'. The problem is that this alternative is less effective – especially as *Chrome* only allows 30,000 rules, far fewer than found in many other popular ad-blocking rules lists. Google does at least promise that it will look at increasing this limit in the future, depending on how it affects

> "CHROME ONLY ALLOWS 30,000 RULES, FAR FEWER THAN FOUND IN MANY OTHER POPULAR AD-BLOCKING RULES LISTS."

performance. This excuse, as Raymond Hill, lead developer of the popular *uBlock Origin* explains, is disingenuous, as web browsing performance is slowed by bloat on webpages (*like adverts?-Ed*) not by the webRequest API.

In a post on GitHub (**http://bit.ly/LXF252uBlock**), he comments "Now that Google Chrome is the dominant browser, it is in a better position to shift the optimal point between the two goals which benefits Google's primary business." Ultimately, it only serves to make alternatives like *Firefox* more attractive.

Google's changes to how Chrome handles ad-blocking has drawn plenty of criticism.

GOVERNMENT

South Korea could switch to Linux

Interior Ministry reveals it is testing Linux – and if no security issues are found, could switch to the OS.

The South Korean government's Ministry of the Interior and Safety announced that it is testing running Linux on government PCs. If the tests are a success, and Linux is rolled out to all PCs used by the South Korean government, it will be another big win for the open source operating system.

Currently, most PCs used by the South Korean government run Windows 7, but with Microsoft's free technical support ending on 14 January 2020, it appears the government is looking at ways to avoid the costs of continuing to use the ageing operating system. However, according to the Korea Herald (**http://bit.ly/LXF252KoreaHerald**) the cost of moving to Linux – and buying new PCs – is expected to cost the government around 780 billion won (£519 million). Savings will be made due to not having to license the open source OS.

Choi Jang-hyuk, the ministry's digital service bureau chief, is also reported as saying that the government hopes it will avoid having to rely on a single operating system. We always welcome news of governments making the switch to Linux and open source software, and avoiding having to rely on a single OS is also wise. After all, Microsoft's decision to end free support for Windows 7, and charge for continued support, has left institutes at the mercy of the company's whims. Switching to Linux will avoid this

But it's not a done deal just yet. The South Korean government is currently testing deploying Linux on private networked devices to check for security risks and to ensure compatibility with websites and software that have been designed to run on Windows.

The South Korean government is contemplating switching to Linux.

Image credit: Getty

SOFTWARE

Unity Engine officially on Linux

For Personal, Plus and Pro licence users.

After years of amazing unofficial developer support for Linux, the *Unity Editor* is now official. According to the **http://bit.ly/LXF252UnityLinux** announcement, "the increasing demand of Unity users in the Film and Automotive, Transportation, and Manufacturing (ATM) industries" has led to official support.

At the moment it's in preview, but it's expected to be fully supported by Unity 2019.3. For the time being, official support will focus on PCs running Ubuntu 16.04/18.04 and CentOS 7 on x86 and x84 architecture, with the Nvidia official proprietary graphics driver and AMD Mesa graphics driver. While the Unity engine is primarily used by games developers, as the announcement reveals this move has been driven by demand from other creative industries. You can download the latest builds via the Unity Hub at **http://bit.ly/LXF252UnityHub**. The Unity team is also keen to hear feedback from Linux users about their experiences with using Unity. If you're using it, let them know on the Unity Editor for Linux forum (**http://bit.ly/LXF252UnityForum**).

If you're looking for a community-developed open source alternative to Unity, now is an excellent time to check out Godot (**https://godotengine.org**). Not only is it open-source and free to use, but it offers a wider array of languages, including GDScript, C# 7.0, C++, Python and more. At GDC 2019, the Godot team held a presentation on the state of Godot in 2019, along with hints at future features. You can watch a video of the presentation at: **http://bit.ly/lxf252godot**.

OPINION

LINUX IN BRAZIL

Mark Filion
Marketing Manager, Collabora Ltd.

"This summer (or rather, this winter, in the southern hemisphere), Brazil will play host to two significant conferences, bringing together Linux enthusiasts and developers alike. Following its first ever presence in Asia in 2018, the Debian Project's annual conference DebConf heads to Curitiba, the capital of the Brazilian state of Paraná. Taking place from 21-28 July, this year's edition of DebConf is the first to be located in South America for more than a decade.

A week later, on 2-4 August and roughly 425km northeast in São Paulo, it's the second edition of Linux Developer Conference Brazil, a conference aimed at giving an international window to the local developer community.

Whether you are a developer, contributor or simply interested in learning more about the Debian Project or the Linux kernel, these two conferences promise to offer a bounty of talks, discussion sessions, BoFs, workshops and more.

DebConf talks are going to be recorded and streamed live, so if you can't attend the conference in person, you can watch them on the Debian website. For more details, visit **https://debconf19.debconf.org** and **https://linuxdev-br.net**."

OPINION

EMAIL HELL

Keith Edmunds is Tiger Computing Ltd's MD, which provides support for businesses using Linux.

"I received a mail this week that said, "I'm looking for a solution to an email alert problem. I get far more email from systems than I do from people". I suspect he's not alone.

For years, the de rigueur way to keep an eye on your Linux estate was to have a multitude of scripts that would inform you of success, failure, concerns and more by email: "Backups ran OK", "237 mails were sent", "/dev/sda3 is 97% full", and so on.

Well, shock, horror: this is 2019, and if you're still relying on those emails, you're doing it wrong. Such an approach is full of problems: there isn't time to action them all; you won't notice if one them doesn't arrive; there's no effective history. Need I go on?

Set up server monitoring. There are lots of open source options, and any is better than none. You could do worse than start with Nagios (*see tutorials in* **LXF243**) if you have fewer than 20 servers. Set the goal: the only mails you want to receive that don't come from a human should come from your monitoring system."

MALWARE

HiddenWasp Malware hits Linux

A nasty trojan has been found that targets Linux systems for remote control.

Cybersecurity company Intezer (**www.intezer.com**) has discovered a nasty piece of sophisticated malware, which it has dubbed 'HiddenWasp'. It targets Linux systems, and as the company explains, unlike other malware that targets Linux for cryptomining or DDoS attacks HiddenWasp is a trojan that is used to remotely control target systems.

Most worryingly of all, Intezer revealed in a blog post disclosing the existence of the malware (**http://bit.ly/LXF252IntezerBlog**) that HiddenWasp is still active and has a "zero-detection rate in all major anti-virus systems." The authors behind HiddenWasp have apparently used a large amount of code from publicly available open-source malware, including Mirai and Azazel rootkit.

INTEZER

Intezer is a cybersecurity company that specialises in open source software.

Unlike Windows malware – which is often carefully written – Intezer says that Linux malware authors do not spend too much effort creating their software, instead picking and choosing from publicly available code. Coupled with the fact that antivirus products for Linux aren't as resilient as on other platforms (again, according to Intezer), it means these sort of threats can go undetected relatively easily.

It's definitely worth visiting the blog to read an in-depth analysis of how the HiddenWasp trojan works, and it also explains how the community should prevent and respond to the threat.

CHIPS

RISC-V dev revs up

Meet the Berkeley Out-of-Order Machine: BOOM.

We always like to see the progress of open-source RISC-V processors, and it looks like RISC-V BOOM, also known as the Berkeley Out-of-Order RISC-V Processor, is coming along nicely. This is a synthesisable and parameterisable open source RV64GC RISC-V core written in the Chisel language. According to the RISC-V BOOM website (**https://boom-core.org**), this is an ASIC-optimised core, but it is also usable on Field Programmable Gate Arrays and was built at University of California, Berkeley in the Berkeley Architecture Research group (**https://bar.eecs.berkeley.edu**).

The BOOM has been developed by the Berkeley Architecture Research group.

Credit: Charlie Nguyen www.flickr.com/photos/7453283@N02/2280342987 (CC BY 2.0)

HARDWARE

Nvidia and Red Hat go for a drive

Graphics driver development is being made easier.

Nvidia and Red Hat have been working closely together for over 10 years to accelerate Red Hat Enterprise Linux (RHEL) workloads on Nvidia GPU-enabled servers, using a software stack powered by Nvidia's CUDA platform. The companies have also been working on improving the user experience to simplify driver development (and other Nvidia software) on RHEL. This collaboration has resulted in new packages for the GPU drivers on RHEL, which are now available as a technical preview.

According to Pramod Ramarao, Senior Product Manager at Nvidia, "the goal… is to improve the user experience of installing and upgrading these drivers on RHEL. By providing better integration of the drivers and RHEL on a technical level, the new packages remove the need to have compilers and a full software development toolchain installed on each system running NVIDIA GPUs" Find out more at: **http://bit.ly/LXF252NvidiaRedHat**.

Distro watch

What's waiting in the free software in-tray?

ZORIN OS 15

Zorin OS is an Ubuntu-based distro that's ideal for people migrating from Windows, thanks to its user interface which is inspired by Microsoft's operating system. The latest version, Zorin OS 15, comes with Zorin Connect (which is based on KDE Connect) for sharing between devices. It also introduces Zorin Auto Theme, which changes the desktop theme into Dark mode at sunset – and back to the normal theme at sunrise. For more information, visit the release announcement at **http://bit.ly/LXF252ZorinOS15**.

Zorin OS is an ideal distro for easing Windows users into the world of Linux.

4MLINUX 29.0

This new release of the lightweight 32-bit distro – which focuses on maintenance, multimedia, miniserver and mystery – is now available. It comes with the Linux kernel version 4.19.41, updated desktop and server packages (including *LibreOffice 6.2.4*, *GIMP 2.10.10*, *DropBox 73.4.118*, *Firefox 66.0.5*, *Chromium 74.0.3729.10*) and much more. The 'mystery' aspect of 4MLinux refers to games, and 29.0 brings improved 3D acceleration for *Quake II*.
Download it from **http://bit.ly/LXF2524MLinux**.

4MLinux focuses on the four 'M's – and the latest version brings some welcome new features for each one.

GPARTED LIVE 1.0.0-1

This milestone release of the bootable CD version of the *GParted* partition manager includes *GParted 1.0.0* and updated packages, among other updates. The Linux kernel has been updated to 4.19.37 and the boot menu has been improved. According to the release announcement (**http://bit.ly/LXF252GParted**), *GParted Live 1.0.0-1* has been tested on *VirtualBox*, *VMware*, BIOS, UEFI and physical computers with AMD/ATI, NVIDIA and Intel graphics.

GParted Live is a version of the GNOME Partition Editor that can be booted from a CD or USB drive.

MX LINUX 18.3

Version 18.3 of MX Linux is out. It's based on Debian 9.9 and features a customised Xfce desktop, and this new version mainly concerns itself with including application updates and bug fixes. The updated applications include *mx-installer* (which is based on the *gazelle-installer*), and improvements to the UEFI boot installation routines have also been added. Find out more at **http://bit.ly/LXFMXLinux**.

This is a hidden Illuminati symbol if ever we've seen one. Which we haven't.

OPINION

DESKTOP FURNISHINGS

Jonni Bidwell has been hiding under his duvet until the weather gets a little warmer, and the internet safer.

"People slated Gnome for getting rid of various things: menu bars, the system tray, minimise buttons, all that is good and sacred. Imagine, then, my surprise when I logged into KDE and discovered not one but two jarring omissions. Firstly, most applications were missing menu bars, these having been demoted to inside a "hamburger" menu (don't get me started). Secondly, file-copying progress was now indicated only by subtle green shading of the Dolphin entry in the task bar.

My fondness for Arch (I use that BTW) is also a matter of public record, so this could be due to teething problems with the new Breeze theme.

The fixes were complicated; first the hamburger had to be removed from the Window decorations, then menu bars had to be re-enabled (per app!) by various shortcuts.

I thought the second issue would be easy, and indeed found and disabled the option to use these awful progress bars (it's in Task Manager settings). But that doesn't give you back the traditional windowed progress bar. Oh, no – that option is hidden away in the Notification settings, an area I never knew existed. Pah."

OPINION

IN MEMORY

Jon Masters is a kernel hacker who's been involved with Linux for more than 22 years, and works on energy-efficient Arm servers.

"Linux is built by many talented human beings around the world. Many of whom have worked together for a very long time, getting on for three decades in some cases. As with any long-lived community, we have had our share of ups and downs. There are tremendously positive stories – people who have achieved international recognition for their work, or met their future spouses. Other times there are life's lesser moments.

It is, then, with some considerable sadness that we all recently learned of the passing of Martin Schwidefsky, long time Linux s390 (mainframe) maintainer, who died in a parachute accident. Like many others, I never met Martin in person, but his positive energy was felt in the community for many years. He was not the kind of person who ever seemed to engage in flame wars, but was more likely to enjoy a polite, technically informed discourse on a wide range of topics, always bringing his best at any moment. He is certain to be missed by many for his kind words, expertise, and mentoring of others. Martin did much of the initial work on the s390 port of Linux, which today has a vibrant community that no doubt will continue the legacy he created. He will be missed."

Kernel Watch

Jon Masters summarises the latest happenings in the Linux kernel, so that you don't have to.

Linus Torvalds announced the 5.2-rc5 (Release Candidate 5) kernel, noting that the release was settling down with "nothing...particularly scary-looking".

Assuming that nothing scary does emerge, 5.2 should be released soon after we go to press, meaning that we will have a full rundown of the new features next issue. This includes support for "pressure-stall monitors" (userspace monitoring of sources of memory pressure that could later slow down a system), the new "mitigations=" one stop shop for controlling security mitigations, and a CLONE_PIDFD flag in the clone() system call to allow userspace (e.g. Android) to monitor processes in a race free manner.

> "It can be painful when the vulnerability mitigation suggests that such threads be disabled"

KVM Address Space Isolation

The past year has seen several hardware vulnerabilities that can impact public and private cloud computing environments.

In several cases, vulnerabilities can leak secrets (keys, and other sensitive data) due to resource sharing within a microprocessor. This includes the CPU memory "caches" used to speed up access to recently used data items. In some processors, the inner most data cache is shared by two sibling hardware threads that appear as if they were separate "logical processors" to the OS but are in fact cunningly splitting the resources of a single processor core. Indeed, in some cloud environments, when you spin up an instance with two "vcpus", you're actually getting one physical CPU core with two of these threads as "CPUs".

It can be painful when the vulnerability mitigation suggests that such threads be disabled (halving advertised CPU resources, even if throughput isn't impacted as much). Hence the need to find mechanisms to isolate secrets shared between unrelated hardware threads so vulnerabilities don't lead to data leaks.

A patch posted by Alexandre Chartre based in part upon earlier work from Liran Alon and others aims to introduce Microsoft-like Hyper-V "HyperClear", creating a separate address space within the kernel for every KVM guest.

Although the guest itself traditionally runs in its own context, the new patches create an additional separate context within the host kernel such that if one thread exits from the guest, it does so in a minimal special environment that doesn't contain secrets the other thread could try to extract. If things work out well, this should be one of the less painful mitigations in terms of performance impact. It's early days, but it is promising. LXF

» ONGOING DEVELOPMENTS

Planning for this year's (invite only) *Linux Kernel Maintainer's Summit* is now in full swing, with Ted Ts'o announcing that the first round of invites would include many of the usual suspects. The *Kernel Summit* will once again be co-located with the *Linux Plumbers Conference 2019* in September (Lisbon). There is usually some overlap and a good opportunity to meet kernel developers in person, even if you just attend the main *Linux Plumbers Conference*. For more info: **http://bit.ly/lxf252event**.

Konstantin Ryabitsev posted a "PSA" (Public Service Announcement) concerning the European GDPR directive and the use of "Reported-By" tags in emails and git commit logs. While he wasn't offering legal advice (and nor are we), he suggested that patch authors consider obtaining permission from those who report bugs to use their name and other identifying information. This lead to various counter arguments about such information already being public.

Yu-cheng Yu posted updated patches for Intel's Control-flow Enforcement Technology (CET). This aims to restrict the ability for programs to be hijacked through stack smashing and other similar buffer overflow attacks.

LINUX USER GROUPS

The intrepid **Les Pounder** brings you the latest community and LUG news.

FIND AND JOIN A LUG

Build Brighton Thursday evening is open night. **www.buildbrighton.com**

Cornwall Tech Jam Second Saturday of the month, alternating between Bodmin and Camborne. **www.cornwalltechjam.uk**

Glasgow Makers and Hardware Hackers Mitchell Library, Glasgow. **https://m.facebook.com/groups/115303729096198**

Huddersfield Raspberry Jam Meet every month at Huddersfield Library, typically on the fourth Saturday of each month. **https://huddersfieldraspberryjam.co.uk**

Horsham Raspberry Jam Parkside, Chart Way, Horsham. **www.facebook.com/hackhorsham**

Leeds Hackspace Open night every Tuesday 7pm-late, open day second Saturday of the month, 11am-4pm. **www.leedshackspace.org.uk**

Medway Makers 12 Dunlin Drive, St Mary's Island, Chatham ME2 3JE. **www.medwaymakers.com**

New Jersey Linux User's Group Last Tuesday of every month, at Panara Bread, 165 Route 4 West, Paramus, NJ. **http://njlnux.org**

PLUG – Perth Linux Users' Group Once a month, at Spacecubed, 45 Saint Georges Terrace, Perth, Western Australia. **www.plug.org.au**

rLab Reading Hackspace Unit C1, Weldale ST, Reading, Wednesday from 7pm. **http://rlab.org.uk**

Teesside Hackspace Tuesday Evenings at Teesside Hackspace. **www.teessidehackspace.org.uk**

The Things Network Reading Walkabout Bar, Reading, 2nd Tues 7pm. **ttnreading.org**

MiXiT 2019

Avez-vous même Linux, bro?

Conferences are binary: they are either community-led or run by corporations. But as with everything, there are exceptions. MiXiT (**https://mixitconf.org/en**) is an event with corporate levels of sponsorship and attendance which still feels like a grassroots event. There were around 1,000 attendees from across Europe inside the Université Jean Moulin Lyon 3, living and breathing the Lyon lifestyle for three glorious spring days.

For two of these days there were talks and workshops from worldwide speakers, including topics such as the 'third space' (the increase in maker/hack/co-working spaces) and creating your own open-source project. The primary language at the conference is French, but there were also a selection of talks and workshops in English.

In the sponsorship hall there were many big-name companies such as Red Hat and Microsoft offering up interesting stalls showcasing their latest tech. Rubbing shoulders with these giants was the Raspberry Pi. Enthusiasts were on hand to demonstrate the small yet mighty computer, and this proved to be an immensely rewarding experience generating lots of interest. But MiXiT is not just about adults, and on the third day there was a special MixTeen event for over 60 children to learn more about computers and coding.

MiXiT is a shining example of diversity and inclusion. Despite our French language skills being rusty, the delegates and team worked to ensure that our time was fantastic. This was also true for one of the youngest delegates, who, at only a few months old, was enjoying their first conference. MiXiT is what a conference should aim to be and we are thankful for their hard work. LXF

MIXIT is a tech conference that keeps the community at its core. Lunch was just as important as talks, of course!

COMMUNITY EVENTS NEWS

PYCON AFRICA
The first ever pan-African Python conference takes place from 6-10 August in the coastal capital city of Accra, Ghana. There will be talks, workshops, community events and the all important 'code sprints' where projects are created and maintained. The event will have talks in the broad subject groups of Python and Data, Web, General Python and Python community. There will also be sessions aimed at younger coders, along with workshops to help educate. More details and tickets via: **https://africa.pycon.org**.

SCRATCH CONFERENCE EUROPE
From 23-25 August 2019 the city of Cambridge, UK, will play host to the 2019 incarnation of the Scratch Conference Europe. This event is aimed at educators who use Scratch to educate and create content for education. This is a hands-on, show and tell, learning event for those passionate about the language. The event is organised by the Raspberry Pi Foundation and promises to be relevant to educators and makers. Get your tickets and more information from **http://bit.ly/lxf252scratch**.

MICRO:BIT LIVE 2019
Celebrating the micro:bit board, this two-day event takes place at BBC MediaCityUK, Manchester on 4-5 October. It's is your chance to get hands-on with the latest ideas, projects and technologies that have been created thanks to micro:bit. Keep an eye on: **https://microbit.org**.

Got a burning question about open source or the kernel? Whatever your level, email it to **lxf.answers@futurenet.com**

Neil Bothwick wonders: FOSS is the point?

Q Large file, small space

How do I generate a very large file, say around 980GB, with the *dd* command? I'm trying to test a quota for an XFS file system under RHEL7. I use the following command as the user oracle, which has the user quota set:

```
$ dd if=/dev/zero of=test.img bs=1024
count=0 seek=980G
```

and it outputs

```
0+0 records in
0+0 records out
0 bytes (0 b) copied, 0,000377899 s, 0.0
kB/s
```

Where is my file?

Michael Pleasant

A You have created an empty file because you have told *dd* to write no blocks, and it has done exactly that. If you run it with `count=1`, you will create a 980TB sparse file, which may or may not trip the quota depending on how it is set up. It creates a sparse file because it doesn't start writing until 980TB into the drive. Sparse files take up only as much space as they need. If you ran:

```
$ ls -l test.img
$ du -h test.img
```

then the first command would show your file as its full size, while the second would show it as occupying very little disk space. Note that this is a 980TB file, because the `seek` and `count` options are given in block units, which you have specified to be 1K. If you want a 980GB file you need to use `seek=980M`, or use `bs=1`. Sparse files are quick to create because you are writing almost no data to them; they are intended to be there for data to be added later, an empty container. Disk image files used by virtualisation software are a classic use case for these. I have had 10 VMs, each with a 100GB disk file, all on a 500GB filesystem because nothing would fill up until the virtual disks started to fill up.

If a sparse file is sufficient to trip your quota limits, this is indeed the fastest way to test it, otherwise you will need to drop the `seek` option and increase the block size to around 4M to get the best performance, but it will still take a while. Don't forget to adjust the `count` setting to suit the new block size.

Q Self-hosted blog

I am interested in starting my own blog. I was originally going to support this on a Windows OS, but I would very much like to stop using Windows entirely. What do I need to host a blog on Linux, and which version of Linux would be best for this?

Scott Howe

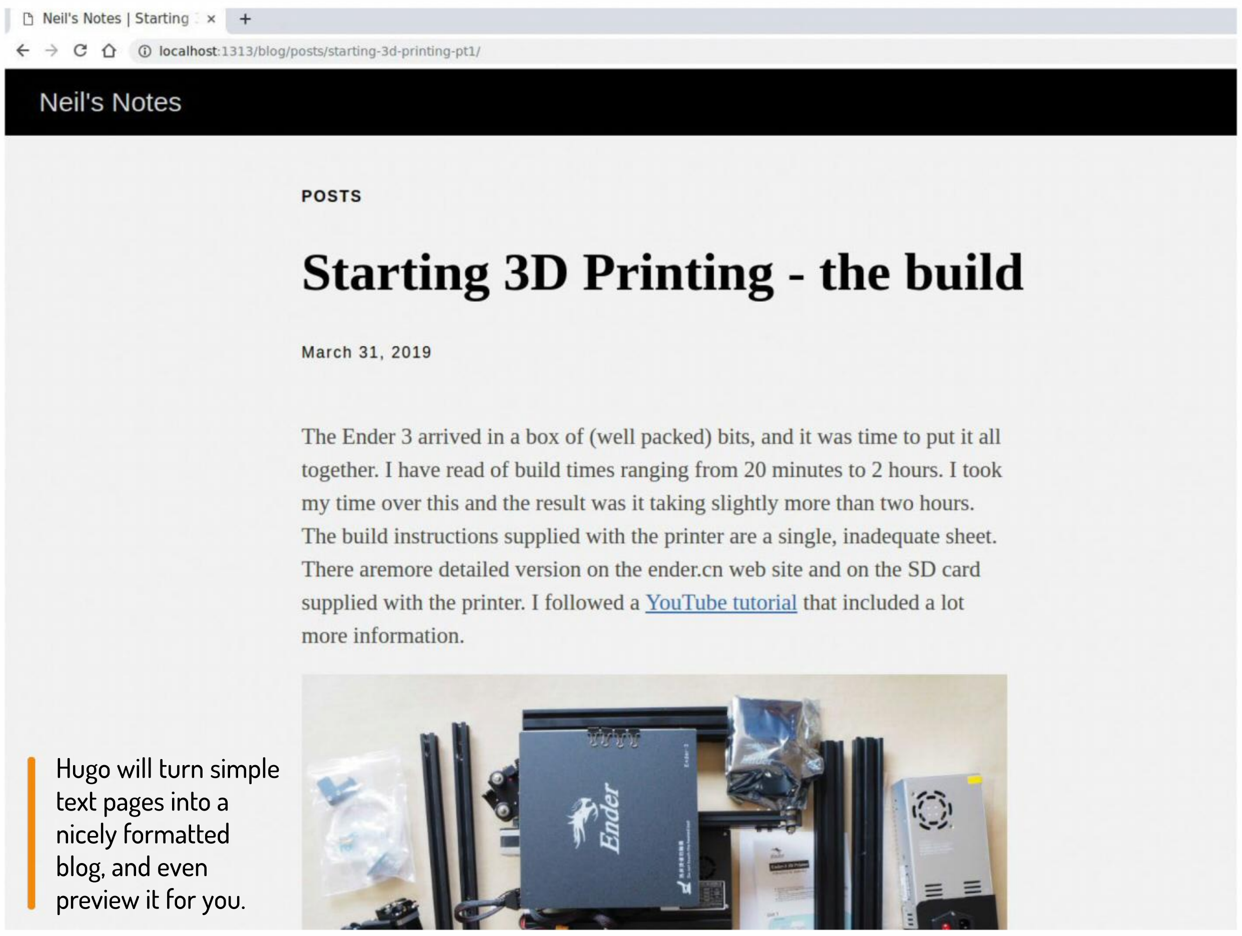
A The choice of distro is purely down to personal taste, as the software needed to host a blog is generic. First of all, you need a web server. The default choice for Linux is Apache, although there are more lightweight alternatives if all you want to do is host a simple blog, such as lighttpd (**www.lighttpd.net**) or Nginx (**https://nginx.org**). Although more heavyweight, Apache is probably the easiest to install, and easier to set up for a blog because most of the software defaults to setting up for Apache.

Then you need something to host your blog. The most common choice for this is probably *WordPress* (**https://wordpress.org**). This is a PHP application but there are frequent security issues; so many sites use *WordPress* that it is a prime target, so it is important to keep this up to date.

A simpler alternative to a content management system like *WordPress* is a static page generator. This takes text files, written in markdown (a simple text markup language) and converts them to HTML for uploading to your site. For a single user blog that doesn't need comments, file uploads or any of the other bells and whistles that the likes of *WordPress* provide, this is often a better choice. There are several such programs: *Jekyll* (**https://jekyllrb.com**) is used by GitHub to generate its user pages, or you can use *Hugo* (**https://gohugo.io**), which I have recently used to deploy a blog. Install it in the usual way then run

```
$ hugo new site myblog
```

This creates a directory called **myblog**, containing several other directories. Follow the instructions to download a theme, either using *git* or by downloading a zip



Hugo will turn simple text pages into a nicely formatted blog, and even preview it for you.

file from **https://themes.gohugo.io** and unpacking it into **myblog/themes**. Then create a text file in **myblog/posts** and write some content. You can use markdown to add links, emphasis or images, then to start a local web server run:

```
cd myblog
hugo server -D
```

Point your browser at **http://localhost:1313** to preview your blog. Any changes you make should then show up in the browser. When you are happy, run:

```
$ hugo
```

to create a complete blog in **myblog/public**. This is a self-contained static site ready to upload to your server. For more information, read the documentation at **https://gohugo.io**.

There are couple of other points to consider when hosting your own site. You need an accessible IP address – either a static address or a dynamic DNS service. You should also consider making your site available over HTTPS by getting a certificate from **https://letsencrypt.org**. See the Apache tutorial in **LXF249** for more.

Grokking grep

Q **I am usually OK with some basic *grep* stuff, but could do with a bit of help here. I need to *grep* a log file for just the last 60 seconds for an error that occurs once in a blue moon. If I find the error I would like to reboot the server but clearly if not, do nothing. I am no expert here; digging around I found some ideas, but nothing I tried worked as I expected. Can you suggest a suitable expression for *grep* here?**

Glenn

A It looks like you want something you can run from *cron* to check if the process has errored since the last run, but *grep*-ping the entire system log – possibly containing tens of thousands of entries – every minute is hugely costly, and may still miss an error if it occurs at just the wrong time. You also need to consider cases like what happens at midnight, when you are looking for times later than the current time, or what happens on daylight saving time change-overs.

A better approach would be to continuously monitor the system log, so each entry only has to be *grep*-ped once. At the most basic level, you can do this by following the log with *tail* and passing it to *grep*. Put it in a short shell script and run it at startup – something like this:

```
#!/bin/sh
tail -n 0 -f /var/log/syslog | grep -q -m 1
"Your error message"
reboot
```

The `tail -f` command follows the given file, outputting any new lines to stdout. By default, it also sends the last ten lines of the existing data: `-n 0` stops this. Then we use *grep* to search for your error message; the `-m 1` (or `--max-count=1`) option tells *grep* to exit after finding one match. So it sits there watching your syslog output until it sees your error message, at which point it exits and your script continues with the next command, `reboot`. For those using systemd's journal, you can replace the `tail` command with:

```
$ journalctl -f -n 0
```

This is a basic but much more efficient and effective way of doing what you want, but there are other options, including various log-watching applications. As the script simply looks for the error and then runs a command, you could have it run any command, or more than one. For instance, you could have it send an email to notify you of the reboot, or maybe a Pushover or Pushbullet notification so your phone beeps when the server reboots.

You may also want to collect and log some information to help you diagnose the cause of the problem and try to avoid the need to reboot so often. If you are doing that, I would recommend adding a `sleep` command before the final reboot, to give the system time to do what you asked.

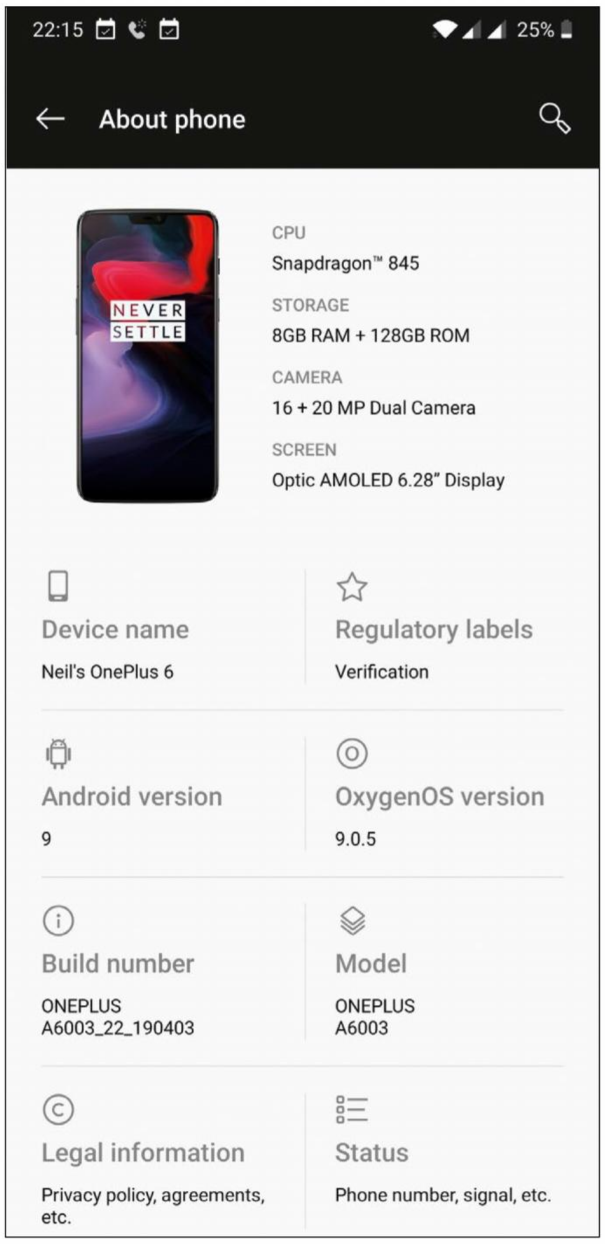
Moving mountpoints

Q **How do I set my computer so that my Android devices 'mount' at /media instead of (or in addition to) /run/user/xxx/mtp:host=yyy? I have a related question: every time I connect my device by USB cable I get different 'mtp:host=' details. Is there some way to control, manage or manipulate this? I'd use predictable details, detect connection, then create a link into /media.**

Martin Kalb

A Most desktop environments and file managers now use *udisks* to handle mounting of removable drives, which includes things like USB sticks as well as media devices. The default behaviour of *udisks* is to mount devices at **/run/media/$USER/$ID**, where $ID is the filesystem label if it has one, or its UUID otherwise. On a multi-user system, this protects devices mounted by one user from being modified by another.

However, this can confuse some software – and users – that expect devices to be mounted under **/media**. There is a way to change this behaviour, by setting a variable in *udisks*' environment to tell it to use a shared mount directory. You set this variable by creating a udev rule. Create



If you want your device to be automatically mounted with a recognisable name, you may need to set the name in the device.

the file **/etc/udev/rules.d/90-udisks2.rules** containing the line

```
ENV{ID_FS_USAGE}=="filesystem|crypto",
ENV{UDISKS_FILESYSTEM_SHARED}="1"
```

There is no need to reboot or restart anything. As soon as you have created this file, which must be done as root, all subsequent mounts of removable devices will be at **/media/$ID**.

Your second issue looks as though it may be caused by the phone. The normal practice is to use the device's name or ID as the mount point – this applies to filesystems as well as other devices.

If your phone is giving inconsistent information, the MTP filesystem will use inconsistent mount points. I would suggest checking your phone's settings, particularly giving it a unique but meaningful name.

Temporarily full

Q **What would happen if /tmp is full? Would the server stop working, or only the applications running on it? Can /tmp be filled by processes in memory, for example?**

Joshua Phelps

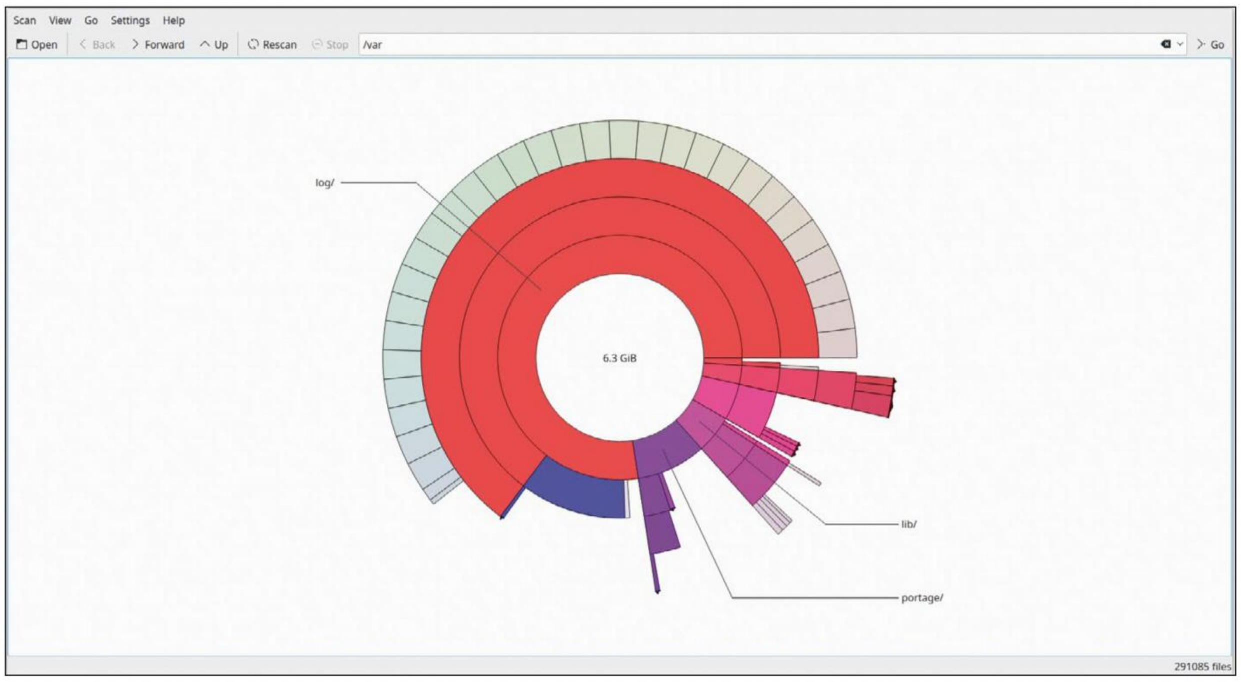
A The **/tmp** directory is normally used to store small, temporary files. In normal use it will not get anywhere near full. For example, on this machine, **/tmp** is currently only one per cent full:

```
Filesystem   Type  Size  Used Avail Use% Mounted on
tmpfs        tmpfs 6.2G  4.0K  6.2G  1% /tmp
```

However, the key phrase here is "in normal use". Any program could write to any directory for which it has permissions, and **/tmp** is usually world-writeable. So a badly behaved or runaway program could keep writing to **/tmp** until it was full. As to whether this would stop anything working, clearly anything that needed to write to **/tmp** would be unable to and would fail in some way, and this would include the program that filled it in the first place.

The effect on other programs would depend on whether **/tmp** was part of the root partition or not. If it is on a separate filesystem then filling it would only affect access to **/tmp**. If it was on the root filesystem, the consequences would be more serious, especially if **/var** was also on the same partition as then nothing would be able to write to log files. Also, you mention a server: many server programs store important data in **/var** and would be unable to function properly if they're unable to write to it.

There are a number of steps you can take to avoid such a situation. If you have a particular program that is using large amounts of space on **/tmp**, can you configure it to use a different location? Moving **/tmp** to a separate filesystem will not prevent it filling up, but it will mitigate the effects of it. It is now popular to use a *tmpfs* for **/tmp**, which stores everything in memory. This makes it faster and means that **/tmp** is automatically cleared on reboot, but does mean that a program that writes large amounts of data to **/tmp** will consume memory.



Keep an eye on disk usage: filling disks completely never has a happy ending, especially if the filesystem contains /var or /tmp.

A typical fstab line for mounting **/tmp** like this is:

```
tmpfs /tmp tmpfs size=50%,noatime,mode=1777 0 0
```

The size argument is optional; `50%` is the default, which means **/tmp** will use 50 per cent of your memory at most. It only uses what it needs, so my 'one per cent full' example is using something like 0.5 per cent of my RAM.

A third option is to use a monitoring program, such as *monit*, to watch your filesystem usage and warn you if it exceeds a certain threshold. This will only be useful if whatever is filling **/tmp** does it slowly or is more of a worry than a repeated event (in which case you should be looking to fix the program that fills **/tmp**). There isn't space to go into the details of setting up *monit* here, but an entry like this in its configuration file should keep you aware of trouble brewing.

```
check filesystem tmp with path /tmp
 if space usage > 50% then alert
```

» A QUICK REFERENCE TO... PIP

Distros have package managers and repositories to make it easy for you to install software, but did you know that some programming languages also do this? Python is increasingly popular on Linux systems and it has a package manager called *pip*. This means that whatever your distro, you can install a Python module with the simple command:

```
$ pip install modulename
```

This installs the module for Python 2, which is still very popular despite Python 3 being over 10 years old. To manage Python 3 modules, simply replace `pip` with `pip3`. The syntax for the two programs is the same (unlike the languages). The main repository of packages, once again distro-independent, is PyPI – the Python Package Index.

As well as the `install` command given above, there is the corresponding `uninstall`, along with `list` to show the packages you have installed, `show` to give more information about those packages and `search`, which you won't be surprised to hear searches the available packages.

Of course, many Python packages are also available from your distro's repositories, so which should you use? Well, using the distro's packages keeps all package management under one roof and minimises compatibility issues – but then again distro packages may not be as recent as those in PyPI, especially if you are using a conservative or older distro.

In some cases, however, there is no choice; if the distro hasn't packaged a module, *pip* is the only way to go.

GET HELP NOW!

We'd love to try and answer any questions you send to **lxf.answers@futurenet.com**, no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (although many suspect Jonni is a robot), so it's important that you include as much information as you can. If something works on one distro but not another, then tell us. If you get an error message, please tell us the exact message and precisely what you did to invoke it.

If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us the **system.txt** file too.

```
uname -a > system.txt
lspci >> system.txt
lspci -vv >> system.txt
```

Mailserver

Write to us at Linux Format, Future Publishing, Quay House, The Ambury, Bath BA1 1UA or **lxf.letters@futurenet.com.**

Credit: Red Hat

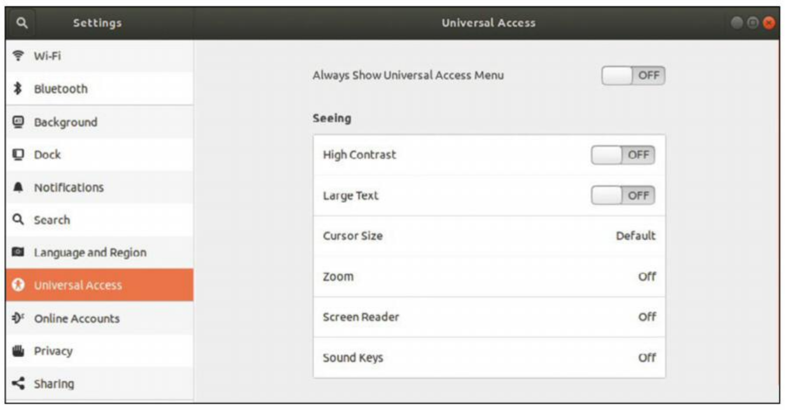
More accessible

I can just imagine it now: your front cover depicting Tux in a wheelchair and his flipper in a sling. Awww!

I just discovered how difficult it is to type with my right hand in plaster following surgery. So I got to thinking whether you could consider an issue of LXF containing an article on accessibility options. There are just a few ideas here, but I'm sure you could improve on my list.

How can I fix 'sticky keys' on my Lubuntu laptop? What is the best voice-assistant for writing and reading text? Is there an easy to use screen magnifier? How to remap the buttons on a mouse? Setting up a touch screen... and so on. I know that some of these options are available to, er, 'other' PC systems, but where is Nurse Tux when we need her/him most? It was a real struggle for me to type this.

Pete, Northumberland UK



Ubuntu and others offer the standard levels of accessibility features.

Neil says

This is certainly something I've always wanted to cover, but unless we had someone with experience or expertise it always felt as if we'd just be paying the topic lip service, rather than giving genuine advice. Hopefully this is something we can dig into in a future issue.

» WIN A SECURE NITROKEY STORAGE 2

Send your thoughts to the *Linux Format* dungeon server at **linuxformat@futurenet.com**, be picked as Letter Of The Month and win* a 32GB Nitrokey Storage 2!

The Nitrokey is worth a cool €149, has 32GB of hardware-encrypted storage, supports hidden volumes, secures your online accounts with one-time passwords, can encrypt emails, files and hard drives and is an open source, open hardware solution. Protected with AES-256 and RSA keys up to 4,096! Learn more at **www.nitrokey.com**.

The Nitrokey is your one-stop security solution.

Font of knowledge

I'm a high-school teacher, and a long-time Linux user. On my personal laptop, an Asus VivoBook, I run Linux Mint 19.1 Cinnamon and it works beautifully. I have two office suites on my laptop: *LibreOffice 6.1.5* and *WPS Office*. Why two? I prefer *LibreOffice* for my own use, and I use *Write* and *Impress* to create assignments and presentations for my students.

I run into problems when I send a document that I created in either suite with the default font to a colleague or a student. Font conversion in *Word* or *PowerPoint* screws up the document, sometimes a little, sometimes a little too much. So my question is this: can I legally copy the Windows TrueType fonts from my Windows 10 computer and install them on

Helpdex

shane_collinge@yahoo.com

* For full terms and conditions see: **www.futureplc.com/terms-conditions**

my Linux laptop? That way, I could change the default font to Calibri, the default font in the latest iteration of Microsoft Office, and avoid the font conversion problems.

Doing this would certainly make life easier, and I realise I could copy the fonts at any time and no one would be the wiser – but since one of the courses I teach includes a unit on ethics, I have to know I'm doing the right thing.

Michael, Florida, USA

Fonts, fonts everywhere but not a single one to print.

Neil says

Good question! You're able to install the Microsoft Core Web Fonts with:

```
sudo apt-get install ttf-mscorefonts-installer
```

The problem is that this requires accepting a Microsoft EULA and does not include Calibri, which there's no ethical way of obtaining. Some point out that it's bundled with the free *PowerPoint Viewer*, but that doesn't strike us as being the intended use. Google has created drop-in replacements that are near-identical – you can install them with:

```
sudo apt install fonts-crosextra-carlito fonts-crosextra-caladea
```

You can read up more on using these at **http://bit.ly/lxf252fonts** – and we also discovered a treasure trove of free fonts at **https://fonts.google.com** along the way!

Arch designs

Would it be possible to include Arch Linux on your next coverdisc, please?

Anthony Bradley, email

Jonni says

When we ran the Arch feature in **LXF249** we did think about including Arch on the disc, but in the end it's not really worth it. The Arch ISO is pretty small, around 900MB. Fair enough, that's still a lot for people without the luxury of fast internet, but even if we did include it all it does is boot to a shell prompt.

You then need to install the thing, which involves downloading several gigabytes of packages! Back to square one, in other words. *(I think he means "No", in other words – Ed)* LXF

» LETTER OF THE MONTH

Blue Hat

Now that IBM has purchased Red Hat, what plans does it have for Fedora and CentOS? Is its interest only in enterprise cloud uses of Red Hat Linux, or will it do any more creative uses of Linux? It has stayed out of consumer computing for quite a while now and a new series of computer products – laptop, workstation, tablet and so on – based upon Fedora Linux would be quite interesting, but I rather doubt IBM is feeling that creative these days.

Maybe it will be very hush-hush about its plans if it's going to do anything exciting. I have been disappointed that HP and Dell have not pushed Linux more aggressively, and it would be nice if IBM stepped into that void with new hardware offerings with Fedora pre-installed here in the US. How about a review of the new laptop that System76 is offering, with larger battery capacity? Does it plan on offering a 13-inch version? I'd also be interested in a review of its desktop/workstation Linux boxes. What features does its in-house Linux distro (Pop!_OS) offer?

Ed Scott, USA

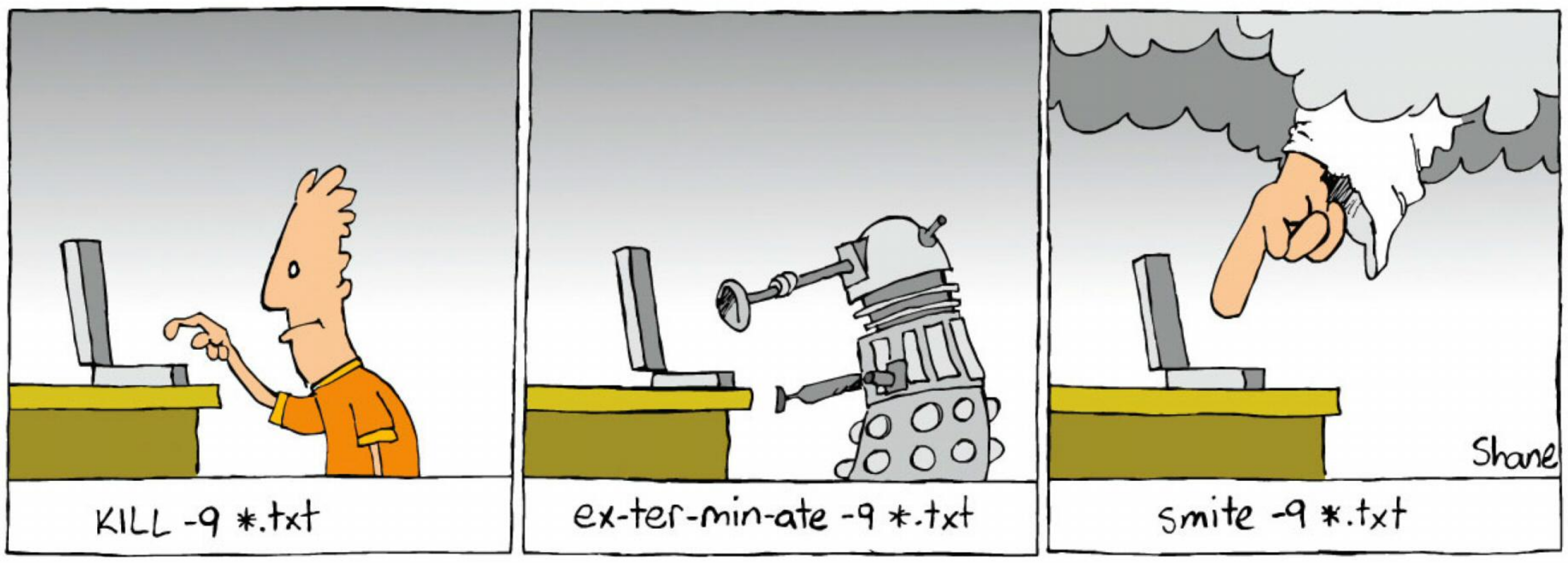
Neil says

I very much doubt we'll be seeing consumer IBM devices again – there's a reason it sold its personal computer division to Lenovo back in 2004. Indeed, it sold Lenovo its x86 server division in 2014 so it could concentrate on services, software and its own Power-based hardware, though hardware has become just a small segment for IBM's overall revenue. Red Hat is there to bolster its service offerings.

I'll step in and defend HP and Dell here; if there were demand for Linux machines they would sell them, and indeed Dell does, as we've covered in reviews in **LXF223** and the interview in **LXF220**. Hopefully you caught our Purism review in **LXF248**, but we should try for more laptop reviews. Basically, if you want Linux on laptops you'll need to support the indie guys making them. As for Pop!_OS, we're looking at it this very issue – see page 46!

Other distros are available, but Jonni would like you to know he uses Arch.

CREDIT: Copyright Judd Vinet and Aaron Griffin.



WRITE TO US

Do you have a burning Linux-related issue that you want to discuss? Write to us at *Linux Format*, Future Publishing, Quay House, The Ambury, Bath, BA1 1UA or email **lxf.letters@futurenet.com**.

MSI GeForce GTX 1650 Gaming X 4G

Nvidia's new GPU brings merely a modest boost in performance, thinks **Jarred Walton**.

SPECS

- **GPU:** Nvidia GeForce GTX 1650
- **CUDA cores:** 896
- **Memory:** 4GB GDDR5
- **Core clock:** Boost: 1860MHz
- **Memory clock:** 2000MHz
- **Outputs:** 2 x DisplayPort 1.4, HDMI 2.0b

The release of the GeForce GTX 1650 was inevitable, considering that none of the other Turing GPUs can fill the role of a budget graphics card. This is likely the final implementation of the Turing architecture, at least at 12nm. The GTX 1650 uses a new TU117 GPU, which is a smaller and thus less expensive variant of the TU116 that powers the Nvidia GTX 1660 (see our review in **LXF251**) and 1660 Ti cards. The key differences relative to the 1660 line are in the memory configuration and the obviously reduced number of streaming multiprocessors.

As expected, the GTX 1650 has 4GB of GDDR5, clocked at 8GT/s – the same speed as the GTX 1660, as well as the previous-generation GTX 1060 cards. Four active memory controllers on a 128-bit bus give it 128GB/s of bandwidth, which is slightly more than the GTX 1050 Ti. Nvidia is typically conservative with its reported boost clocks, with most cards running well above the given speed. The 'stock' GTX 1650 has a boost clock of 1665MHz. That's less than the GTX 1060 cards, but roughly 50 per cent faster than the GTX 1050. The GTX 1650 is also designed to run without a six-pin PCIe power connector – though factory-overclocked cards, like this MSI GTX 1650 Gaming X 4G, have higher clock speeds thus and require a six-pin PEG connector.

Nvidia claimed the GTX 1650 would be up to twice as fast as a GTX 950, and 50 per cent faster than the GTX 1050 – and that was probably a fair estimate, especially since both of those cards only have 2GB VRAM. Overall, the GTX 1650 lands right about where expected. It's faster than the GTX 1050, by over 70 per cent, at 1080p ultra settings. Again, a lot of that is due to the 1050's limited VRAM, as the GTX 1050 Ti is much closer: the 1650 is about 30 per cent faster. The GTX 1060 3GB leads by nine per cent at 1080p ultra, and the AMD RX 570 4GB is in a similar position.

All of which means it could make sense as a cheap upgrade for an older PC, but it's not necessarily the best performance or value you can find right now. From a pure value proposition, especially with the discounted Nvidia GTX 1060 3GB cards still hanging around along with AMD's RX 570 and 580, the GTX 1650 comes up a touch short. For a little more cash you'd even get 8GB of memory with AMD.

The GTX 1650 isn't a bad GPU by any means. It can hit 60fps at 1080p medium to high quality in most games, and it's affordable. Just make sure you understand what you're getting, because if you're looking at building a budget gaming PC, you'll almost always be better served by spending a bit more money. For a compact build or something used for light to moderate gaming, the GTX 1650 is worth a look. LXF

VERDICT

MANUFACTURER: MSI **PRICE:** £160
WEB: www.msi.com/graphics-cards

FEATURES	8/10	EASE OF USE	9/10
PERFORMANCE	6/10	VALUE	7/10

Costing more than a raft of older cards while also performing more slowly leaves the GTX 1650 looking for a reason to exist.

» Rating 7/10

Robolinux 10.5

Mayank Sharma isn't too gullible in real life, but has an uncanny tendency to fall for distros that then don't deliver on the claims they make.

IN BRIEF

A customised Ubuntu-based distro that ships with multiple desktop environments. It pitches itself to users who want to migrate away from Windows by helping them roll their existing Windows XP, 7 and 10 installations into a virtual machine. Another highlight is its one-click installers for popular apps, to help shield the app installation process from new Linux users.

Never one to judge a book by its cover, we let slide the fact that Robolinux's website is very messy and a nightmare to navigate. Sure, we appreciate the fact that donations are key to sustaining the open source ecosystem, but we think this project goes a little overboard. Its developer's heart is in the right place; he isn't just seeking donations for himself, but that's no excuse for the poorly designed website.

Robolinux has three active branches: Raptor 8, based on Debian; Raptor 9, based on Ubuntu LTS; and Raptor 10, based on Ubuntu. Each branch has a bunch of releases for different desktop environments, including Cinnamon, GNOME, Xfce, MATE and LXDE, for a total of 15 ISO images. Once you do manage to navigate your way to the downloads page, you find that they're actually hosted on SourceForge.

The project's poor taste in user interface design extends from the website to the desktop, with gaudy wallpaper and silly desktop icons.

The ISOs are hefty compared to other distros, ranging between 2-4GB. There's obviously no shortage of pre-installed apps and you can get more using the distro's unique one-click installers. When we last looked at the distro in **LXF227**, the project was charging $10 (about £8) for them. Unless you shelled out the money, you couldn't install any of these apps. That's changed now and you can grab the installers from the project's website. The apps are essentially just shell scripts that use *apt* to get the apps.

We have two issues with Robolinux's approach. The first is that while they do just take a single click to install, it takes dozens to grab the scripts from the project's website. Secondly, the project does a disservice to the open source community by claiming the scripts will help save users several hours spent resolving dependencies if they were installing the apps manually. They don't make any mention of dependency-resolution binary package management tools like *apt*, which is what they themselves use in the scripts.

Then there's the other highlighted feature of the Robolinux project, dubbed Stealth VM. Together with another script descriptively named C-Drive-to-VM, this helps you move your physical Windows installation into a virtual machine. Again, both scripts were available only after the payment of a small fee, but are now available for no cost. Peer under the covers and the process is pretty much the standard affair. C-Drive-to-VM gives you a couple of freeware utilities to resize your Windows partition and fold it into a VHD file. After moving the VHD file to your Linux installation, use Stealth VM to first convert it into VDI and then import into a *VirtualBox* VM.

We aren't fans of the project shielding this information from its users and projecting it as a novel approach.

The distro also appeals to privacy-conscious users by inviting them to use a script to route all its traffic via a VPN, which is also Robolinux's third distinguishing feature. This too is a disappointment as the service it relies on requires users to have a paid account, which is a fact that is conveniently missing from Robolinux's website. Once again, we aren't against paying for a service, but it's the project's lack of clarity that upsets us.

Lastly, for a distro that targets new users Robolinux doesn't have a public forum board. Instead guaranteed support is doled out by the developer himself – for a fee. There is an option to email your tech support queries for free, but that's no excuse for not hosting public forum boards. The amount of documentation on the project's website is also negligible and isn't properly arranged inside a dedicated menu, which make it hard to access.

Sure, projects need to sustain themselves – but the tactics employed by Robolinux not only make it look shady but also reflect poorly on the larger open source community as a whole. LXF

VERDICT

DEVELOPER: John Martinson
WEB: https://robolinux.org
LICENCE: GPL and others

FEATURES	5/10	EASE OF USE	4/10
PERFORMANCE	4/10	DOCUMENTATION	4/10

Robolinux is one of the shadiest distros we've seen in a while. There really isn't anything to see here..

» Rating 4/10

Emmabuntüs DE 2 1.04

Not a big fan of the kitchen-sink approach, **Mayank Sharma** runs into one distro that's put in enough effort to perform surprisingly well.

IN BRIEF

Emmabuntüs owes its name to the French Emmaus charitable movement and the project has various distros based on Debian and Ubuntu. They use the lightweight Xfce desktop to ensure a smooth user experience on a wide spectrum of machines, including older underpowered ones. At the same time, it wants to be beginner-friendly as well in order to appease first-time users.

The fact that this distro which is designed to work with reconditioned computers ships as a 3.3GB ISO image feels a little strange. But then its developers had to take into account the possibility that a majority of its users would not have access to the internet. This explains why it includes virtually all popular open-source apps as well as several commonly used proprietary ones. In our experience, the 'everything and the kitchen sink' philosophy doesn't augur well and only ends up confusing the user.

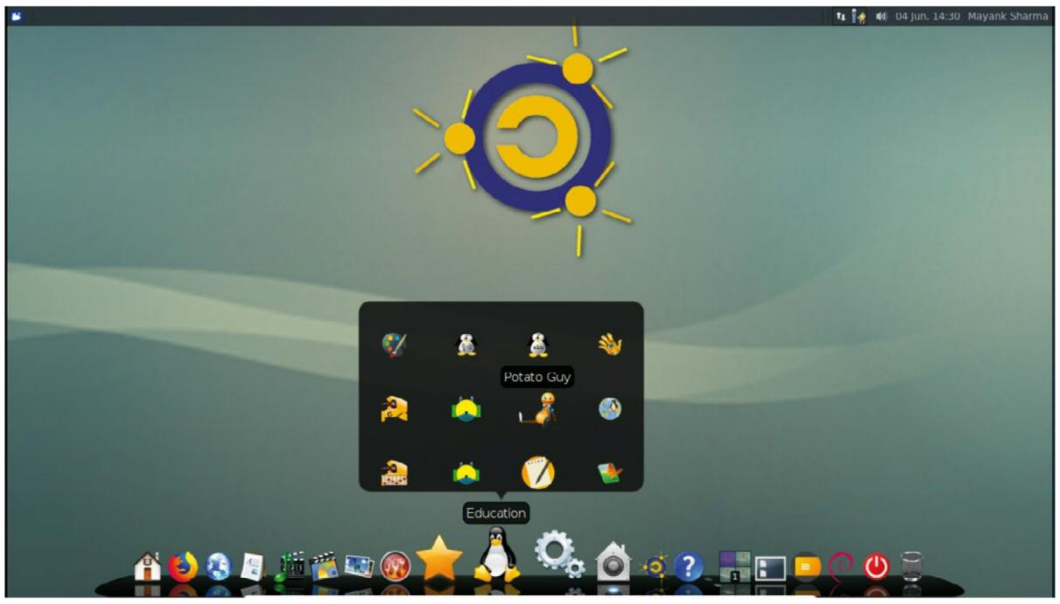
The Emmabuntüs project maintains several editions and the one we are reviewing is based on current stable Debian Stretch v9.9, featuring a customised Xfce desktop. Boot into the Live environment and you're greeted with a first-boot wizard to help you setup the desktop. One of the objectives of the distro is to make itself attractive for first time users, and it does this by enlisting the help of Cairo-Dock. The first-boot wizard helps you switch between three preconfigured configurations of the dock. There's the default Simple version for new Linux users, a Basic version meant for a younger audience and a Full version that exposes all the bundled apps.

A slightly tweaked version of the wizard shows up after you've anchored the distro to your hard disk. This has an additional screen to help you personalise the desktop, with things like changing the wallpaper and switching to Xfce's Whisker menu instead of the default traditional one. It's followed by the proprietary software dialogue box, which shows a list of proprietary apps and codecs including *Skype* and *TeamViewer*. Except for the Flash plug-in, all the other proprietary apps and codecs are bundled with the distro itself and you don't need an active connection to the net to install them.

After installing the non-free bits that you selected, the final dialogue box enables you to remove the language packs that you don't require. By default, the distro installs support for French, English, Spanish, Italian, German, Portuguese and Arabic.

Clever selection

The distro's applications menu is loaded to the brim and would end up confusing new users. That's where the distro's clever Cairo-Dock configurations come into play, as they limit the number of apps and eliminate duplicates for a smoother workflow, especially for first time users. In case you still need to flesh out the distro, in addition to binary apps that you can pull in using *GNOME Software* the distro is also equipped to install Flatpaks.



The distro intends to extend the life of a computer since it says it requires 240 kg of fossil fuels, 22 kg of chemicals, and 1500 liters of water to make one!.

Another highlight of the distro is its welcome app, which includes various options to help users orientate themselves with the installation. You can also use this app to access common administration utilities like boot-repair, device driver manager, NTFS config tool and more. The custom dialogue has several interesting options such as a button to install the LXDE desktop, and various utilities to configure a printer.

In terms of performance, the distro does well and idles at under 350MB memory used. This means you can technically use it on a machine with less than 1GB of RAM, but expect pretty long start-up and app launch times with that configuration. On a fairly recent machine though, it won't give you any reason to complain.

All things considered, while it does cater to desktop users it's unfair to pit Emmabuntüs with regular desktop distros like Ubuntu and Fedora. It's a niche distro built with a specific purpose, which it serves well. Instead of a random assortment of apps, the distro is a purposeful assembly of the right components, which also makes it a wonderful option for introducing new users to the world of Linux and open source. LXF

VERDICT

DEVELOPER: Emmabuntüs Collective
WEB: https://emmabuntus.sourceforge.io
LICENCE: GPL and others

FEATURES	8/10	EASE OF USE	8/10
PERFORMANCE	8/10	DOCUMENTATION	8/10

If you look past its minor peculiarities, you'll discover a well put-together distro that'll appeal to all users.

» **Rating 8/10**

HexagonOS 1.0

A sucker for home-brewed apps, **Mayank Sharma** gets lured into a new distro that's headed in the right direction, but still has a long way to go.

IN BRIEF

The first major release of the 64-bit only distro that's based on the Ubuntu 18.04 LTS release. Developed by an Italian startup, it includes several custom apps and utilities. HexagonOS is designed to be lightweight and aims to aid with common desktop tasks with its customised and intuitive Xfce desktop environment.

Generally we steer clear of new Ubuntu-based distros, but HexagonOS appealed to us with its home-brewed apps. After all, it's logical to assume that anyone who has put in the time to write custom apps must have put in the effort required not to ship just another Ubuntu clone. HexagonOS doesn't disappoint us for being an Ubuntu clone. We are instead let down by the distro's liberal use of the 1.0 versioning. Why would anyone want to subject their project to a reviewer's wrath, knowing very well that their distro is so full of rough edges that it might cut someone?

We don't want to call it as such, but in its current state, the distro is little more than an aesthetically pleasing Ubuntu clone.

HexagonOS is produced by a small Italian startup. It boots into a beautifully crafted Xfce desktop that sports the Arc icon theme, with flat icons that look elegant and modern. There's a smattering of hexagons all over the desktop, from the middle of the wallpaper to the Whisker menu icon, but it doesn't look out of place or forced. The desktop also has the Plank dock. The developers refer to it as the HexagonAutoDock, which apparently solves an issue with Xfce and the Plank dock, but they don't go into details about the issue and their solution.

Then there's *UBackup*, which is pitched as a simple **home** folder backup tool. True to its description, the app only has a single Start button, and simply compressed the contents of the entire **~/** folder into a zip file inside the **home** folder itself. It doesn't prompt for any options, which makes it fairly simple but also very rigid and inflexible, which isn't how we like our backup apps.

Déjà Dup is our gold standard for user-friendly backup tools that are intuitive for new users and offer enough tweakable options for those who want to do customised backups. *UBackup* simple places a zip file in your **home** folder which is nothing more than a compressed dump of your **home** folder. Subsequent backups aren't incremental in nature and will replace the previously created zip file if you haven't moved it somewhere else.

It gets worse…

The other thing that ticked us off was the lack of any restoration option. There is no option in *UBackup* or inside the compressed backup file itself to restore it partially – or even completely, for that matter.

In fact, since it replaces previous backup files, if you happen to take one after accidentally removing a file that file will be lost for good, since the app will overwrite the good backup file. *UBackup* might be a work in progress, but it shouldn't have been placed in a 1.0 release in its current form.

Finally there's *HexagonCenter*, which is the project's solution to helping new users flesh out their installations. It brings up a list of five app categories, each of which has a number of apps. With a single click, the app installs all apps that have a toggled box.

While its heart is in the right place, the app has a number of implementational issues. For one, it asks you to stop using the computer while it is working on installing the apps. There isn't any indication of progress, though you get notifications when the installation starts and ends. Lastly, the app also doesn't give any indication of previously installed apps.

The issues (and our frustration) are compounded by the fact that the project has nothing more than a single page of documentation. We have in the past reviewed distros with non-English documentation, but we don't downrate them for this reason, thanks to Google Translate. HexagonOS, however, has just committed harakiri by jumping the gun on a 1.0 release, which is frankly loaded with underdeveloped apps that do more harm than good. LXF

VERDICT

DEVELOPER: PyrosSoftware
WEB: https://hexagon.pyrossoftware.com
LICENCE: GPL and others

FEATURES	6/10	EASE OF USE	6/10
PERFORMANCE	8/10	DOCUMENTATION	1/10

The numbers don't express our disappointment with the project. We'll ask you to stay clear since there isn't much to see here except for the foundations of an upcoming distro.

» Rating 6/10

OpenSUSE 15.1

This project has been in a state of unrest for some time now, and **Mayank Sharma** wonders if that has left a mark on the distro.

IN BRIEF

OpenSUSE is to SUSE Linux Enterprise (SLE) what Fedora is to RHEL. Once a leading RPM-based distro, openSUSE has lost ground to its peers of late due to the constant change of its corporate ownership, which invariably brings with it questions on its continuation. With the release of openSUSE 15 last year, the distro realigned its version numbers with SLE 15.

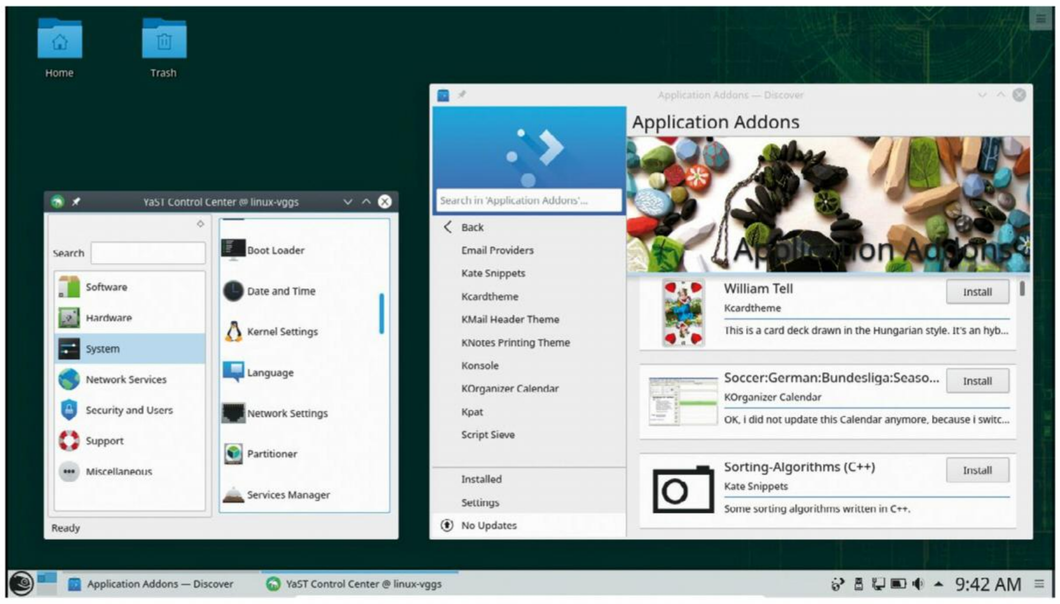
We last reviewed openSUSE more than three years ago and the distro has had some interesting updates and changes since then. It has reorganised, and starting with the 15.0 release last year, SUSE's enterprise version and openSUSE are now being developed in tandem from the ground up.

The openSUSE distro maintains two main branches. There's a rolling release dubbed Tumbleweed, pitched at developers and experienced users. Leap is the regular distro that comes out once a year, and because of its alignment with SLE will not get any major architectural changes for several years, unlike its closest competition Fedora and Ubuntu. This apparent sluggishness helps openSUSE Leap pitch itself as a much more stable option that claims to be better suited for running servers and enterprise desktops.

The Leap flavour that we're reviewing is available in two formats. There are a couple of Live installable mediums for Gnome and KDE, as well as an all-encompassing install-only medium. OpenSUSE now sports a new installer that has everything we want from a distro installer; it's intuitive to operate, comes with reasonable defaults, and offers enough tweakable parameters for advanced users to mould the installation.

One of the most crucial elements of any installer is the partitioner, and openSUSE's has been rewritten to be more dexterous. It offers an interesting option called System Roles, which are predefined use-cases to help setup the system for a particular purpose.

In addition to Gnome and KDE desktops and a Server option, the 15.1 installer has a new role called Transactional Server. This is different from the server role in that it uses a read-only root filesystem and comes from the Kubernetes on SUSE project, Kubic. The pitch for this feature is that it includes an update system that applies updates as a single operation, which makes it easier for users to do Btrfs rollbacks for multiple packages.



YaST features several improvements, such as support for HiDPI displays, and many of its services have been overhauled to use features offered by systemd.

Incremental updates

There have been several changes and improvements for desktop users as well. For instance, the distro has finally moved to the *Firewalld* firewall management tool. Also, while 15.1 ships with the 4.12 kernel, the developers have back-ported graphics drivers from the 4.19 kernel to enable the distro to support newer graphics hardware, particularly that based on the AMD Vega chipset. Administrators who want to roll out openSUSE on multiple computers will appreciate the *AutoYaST* mechanism for mass deployments that features improved usability with more useful profiles and other changes. The installation in itself is a pretty standard affair. Like most leading distros, openSUSE gives equal weight to both KDE and the Gnome desktop, and both perform well and are equally stable. The *YaST* configuration tool remains one of the unique features of the distro and you can use it to influence all aspects of the installation. Besides the package management rolled into *YaST*, the distro includes a graphical app store-like app called *Discover*. In addition to the usual app categories, this has the option to install available app add-ons as well.

One oddity we noticed is that while the distro includes the *VLC* media player, it isn't of much use straight out of the box as the distro doesn't ship with codecs to play the files in the most popular formats. To install the necessary codecs you'll have to pull them after enabling the Packman repository.

This process, however, won't be obvious to new users unless they spend some time reading through the project's documentation. The developers should seriously consider including useful information like this within the installation itself. LXF

VERDICT

DEVELOPER: OpenSUSE Project
WEB: www.opensuse.org
LICENCE: GPL and others

FEATURES	8/10	EASE OF USE	8/10
PERFORMANCE	7/10	DOCUMENTATION	9/10

OpenSUSE Leap is a good option for both servers and desktop installations. Both get new and useful features.

» Rating 8/10

SUBSCRIBE!

Sign up today and get your

Anker PowerWave Fast Wireless Charging Pad

YOUR GIFT!

WORTH £31.99

Don't miss out, subscribe now!

IT'S EASY TO SUBSCRIBE!

https://myfavouritemagazines.co.uk/lin/ankert

Call: **0344 848 2852**

Overseas subs on p67

» PLUS: Exclusive access[1] to the *Linux Format* subs area!

1,000s of DRM-free PDF back issues and articles! Get instant access back to issue 66 (May 2005) with tutorials, interviews, features and reviews. At **linuxformat.com**

1) Only available to MyFavouriteMagazines.co.uk subscribers.

» CHOOSE YOUR PACKAGE!

PRINT EDITION

» Every issue comes with a 4GB DVD packed full of the hottest distros. Plus **Anker PowerWave**

£36 **(16% saving)**
Every six months by Direct Debit

£66 **(21% saving)**
Annual bundle by Direct Debit

PRINT + DIGITAL

» Get both the print & digital editions + DVD, for one low price. Plus your **Anker PowerWave**

£56 **(36% saving)**
Every six months by Direct Debit

£85 **(51% saving)**
Annual bundle by Direct Debit

*Terms and conditions: This offer is only available for new UK subscribers. Gift is subject to availability. Compatibility, 10W Fast Charge Mode using QC 2.0/3.0 Adapter: Samsung Galaxy S9+ / S9 / S8 / S8+ / S7 edge / S7 / S6 edge+ / Note 8. 7.5W Charge Mode, using QC 2.0/3.0 Adapter: iPhone X, iPhone 8 / 8 Plus. Standard Charge Mode using 5V/2A Adapter: Samsung Galaxy S6 / S6 edge, LG G7 / V30+ / V30 / V35, Sony XZ2, Pixel 3 / Pixel 3XL (The phone's screen will display the message 'Charging slowly'.) Do not use Apple's 5V / 1A stock charger or a computer's USB port to power your Anker wireless charger. iOS 11.2 required to access 7.5W fast charging for iPhone. Quick Charge 3.0 wall adapter required for Fast Charge Mode. Metal attachments, phone grip or credit cards will interfere with charging. Cases with metal inside, such as Pitaka, Evutec and Spigen QNMP, will trigger the foreign object detection and interrupt charging. Input: 5V-2A/9V-2A/12V-1.5A. Output: 5W/7.5W/10W. Please allow up to 60 days for the delivery of your gift. In the event of stocks being exhausted we reserve the right to replace with items of similar value. Prices and savings quoted are compared to buying full-priced print issues. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Payment is non-refundable after the 14-day cancellation period unless exceptional circumstances apply. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) or are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: www.bit.ly/magterms. Offer ends 31 August 2019.

Roundup

Equinox Desktop Environment (EDE) » Lumina » LXQt » Moksha Desktop » Openbox

Shashank Sharma

By day Shashank is a New Delhi trial lawyer, but by night he's an open source vigilante!

Lightweight desktops

With his gym membership going unused, **Shashank Sharma** decides to cut down the flab on his desktop instead.

HOW WE TESTED...

We installed all the desktops inside an Arch installation. Most of them are in one of the officially supported repositories, though some are in the community-powered Arch User Repository (AUR).

We've rated the desktops on various fronts, such as their availability and cache of default apps. Malleability is also another important criteria when choosing a desktop; an environment that ships with loads of tweakable settings will score higher than one that has a limited number of modifiable parameters, as people usually swap their defaults DEs because of a rigid configuration. Just as important is the availability of plug-ins and add-ons. These give you the option to extend the usability of your environment depending on your requirements.

The most important criteria for this *Roundup*, however, is the usability of the desktop. Any desktop that doesn't fare well in this test wouldn't be of much use, despite achieving high scores on other fronts.

Initially at least, it'd seem counter-intuitive that anyone would want to change the default desktop environment (DE) of their distribution. After all, the leading distributions spend a considerable amount of time fine-tuning the user experience. Some are even actively involved in the development of the DE to ensure that it meets their expectations. However, the default environment, pretty much like the factory installation, is designed to suit most common workflows.

Despite all their refinements and functionality, some DEs are just a hindrance, especially for tasks that need the least amount of graphical oomph. Then there are those of us who have interacted with computers all their life with a DE that conforms to the traditional desktop metaphor, and so don't really see the advantages of newfangled desktop shells. Another group of users disgruntled with the default DE are those computing on under-powered machines which lack the resources to satiate the needs of the resource-guzzlers.

If you identify with any of the use cases we've just outlined, this *Roundup* is for you. We'll look at some desktops that have a minimum footprint but will still enable you to get along with your tasks.

Availability

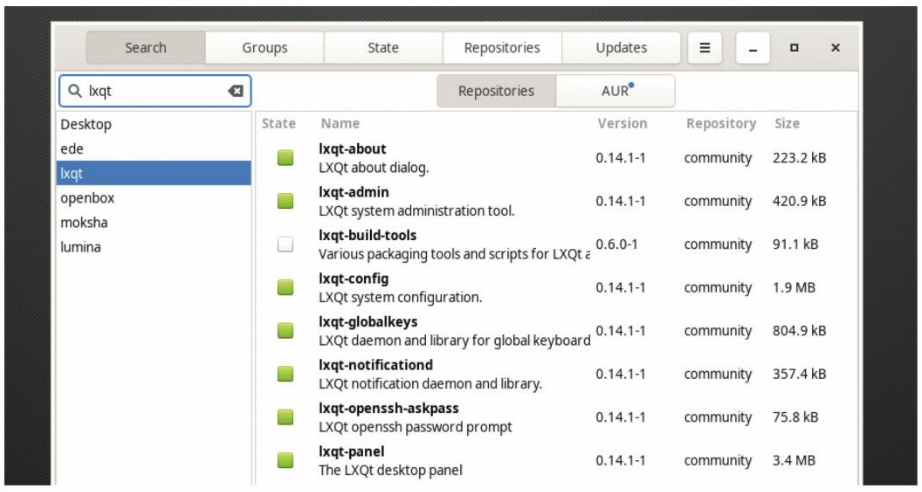
Can you find it in a repository near you?

With the exception of a couple, all the desktops in this *Roundup* are available in the repositories of mainstream desktop distributions. Arch, however, leads the pack, in that it gives you access to all the options via its official and community-supported repositories. You'll also get good mileage with Fedora, thanks to its Copr build system which complements its repositories.

Similarly, Ubuntu with its PPAs enables you to get your hands on a wide number of DEs. One word of advice though: the quality of packages in the community-maintained repositories varies widely and ranges from anywhere between rock-solid to barely usable. You might not get the best experience using packages from a community-supported repository as compared to the official ones.

LXQt and Openbox are the two options that are universally available in almost every Linux distribution. LXQt maintains an extensive list of projects it supports officially on its website, and you can install it on Fedora using a Copr repository. Openbox too is universally supported. We couldn't bring up the right-click applications menu using the package in the repositories of Fedora 29, but that was easily (*oh really?–Ed*) resolved by downgrading the python-pyxdg package.

Lumina is primarily developed for the BSDs, but has also been ported to some Linux distributions. The project's page lists a few distributions that have pre-built packages, including Debian, Gentoo and Arch. The desktop is also available via a Copr repository for Fedora, but it doesn't perform well and crashed during virtually every session. Ubuntu users can find guides to take them through the process of compiling it from source, which isn't too cumbersome.



We'll advise against installing several desktops inside a single installation. Not only will it needlessly clutter the application menus, some don't play well with the others.

Equinox Desktop Environment (EDE) hosts installation instructions on its official wiki and covers Arch, Debian, Slackware, Alpine and a couple of BSDs. The project itself lists these packages as experimental and advices you to compile EDE from source for the best user experience.

The Moksha desktop is a fork of the Enlightenment (E17) desktop. Again, it's the default only on Bodhi Linux and we couldn't get it to work on any other distribution besides Arch. There is a Copr repository but its last build failed for Fedora 25.

VERDICT

EQUINOX DESKTOP	**5/10**	**MOKSHA DESKTOP**	**5/10**
LUMINA	**7/10**	**OPENBOX**	**8/10**
LXQT	**9/10**		

Desktops designed for a particular distribution don't work well on others.

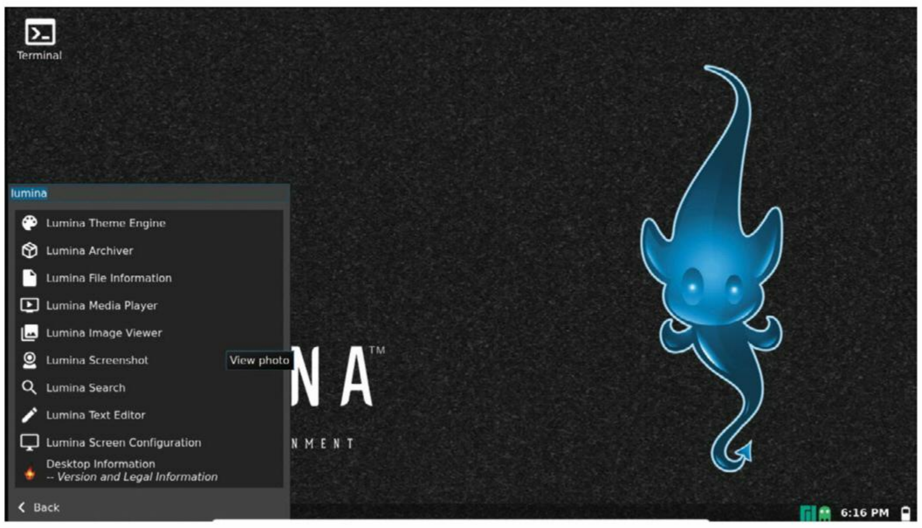
Default apps

What's on offer?

One of the ways some desktops help you conserve resources is to offer matching lightweight apps for common desktop tasks such as file management, text editing, image viewing and so on. Others leave it to the user to pair the lightweight desktop or window manager with third-party lightweight apps.

EDE belong to the latter category. The goal of the desktop is to provide a desktop environment only, and you will have to add even the most common apps like a file manager. Openbox similarly concentrates on being a lightweight window manager that you can complement with third-party light apps. It's the same story with Moksha. Since it was designed to ship with Bodhi Linux, the desktop borrows the few essential apps that are bundled with the distribution.

The other two options are a lot better in terms of bundled apps. Admittedly, you won't find apps such as web browsers, email clients and office suites in Lumina, but the desktop does bundle a handful of utilities that are written specifically for the project. These include an archiver, a scientific calculator, a configurator, a PDF viewer, a text editor, a media player and a file manager. LXQt too features a small cache of optional apps which add extra capabilities to the desktop. There's an image viewer and screenshot tool, a lightweight terminal emulator, a file archiver, a process manager, a file manager and desktop icon manager – which is essentially just a Qt port of *PCManFM* – and more along those lines.



Lumina has a good collection of lightweight apps which nicely complement the desktop itself.

VERDICT

EQUINOX DESKTOP	**1/10**	**MOKSHA DESKTOP**	**2/10**
LUMINA	**7/10**	**OPENBOX**	**1/10**
LXQT	**8/10**		

You'll like Openbox's clean slate if you want to build your own environment.

User Experience

Are they a boon or a bane?

Lightweight or not, a desktop environment is one of the most crucial elements of a distribution. Regardless of the features or application set with which a distribution ships, oftentimes it's the look and feel, behaviour and responsiveness of the desktop environment which help you decide on a distribution.

Perhaps this explains why most mainstream distributions spend a considerable amount of time polishing the desktop environment and ensuring a rich and rewarding user experience.

But all of those careful design strategies go out the window when you swap the default environment with a lightweight alternative. That said, while the primary objective of these desktops is to save resources, it doesn't mean they totally ignore the usability aspect.

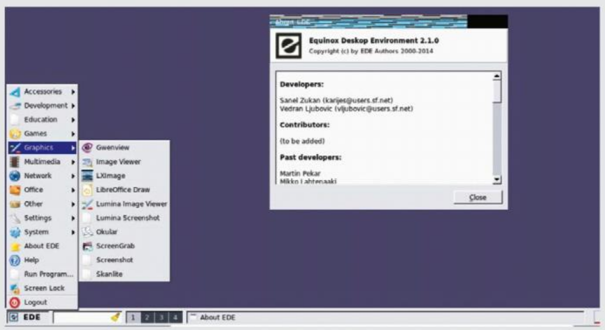
After all, no one wants to use a desktop that takes a negligible amount of memory but which is extremely cumbersome to operate.

Equinox Desktop 7/10

EDE isn't the easiest desktop to install. If you aren't running Arch, chances are you'll have to manually compile it from source.

The desktop has a retro feel to it and starts with an old-school splash screen. EDE adheres to the classic desktop metaphor, with a status bar at the bottom of the screen and the applications menu at the extreme left. You use the text box besides the applications menu button to run a command, which can also be used to quickly launch apps. There's also a pager to switch desktops.

The right-click context menu offers the traditional options and helps place icons on the desktop and change the wallpaper. You can tweak other settings with the desktop configuration app, which provides a bunch of tools to customise other basic elements. Advanced options can only be set by editing text files, which robs the desktop of some usability points.



Lumina 7/10

From our experience, getting this desktop to work on Linux is a really cumbersome process. It's unstable and behaves abnormally atop most distributions, with frequent crashes and weird graphical artifacts pasted across the desktop.

Instead of a traditional splash screen, Lumina instead displays a quote. The desktop has a clean layout, with a status bar at the bottom. The right-click context menu has a number of useful options, including an applications menu. This is a good thing™ because the default applications menu takes a little getting used to. In the default view it only displays favourite apps, and it takes an additional click to get to the full list of installed apps.

There's also a textbox to help you zero in on the app you wish to launch. Besides the apps, you can also launch the file manager and the Lumina configuration utility from this menu.

Help and support

What to do you when you're stuck?

EDE's official wiki has instructions on how to compile it for various desktops – but the majority of the documentation is geared towards helping developers contribute to the project. Also, unlike some of its peers in this *Roundup*, EDE gets very little coverage on other websites besides its own, primarily due to its slow rate of development.

Lumina is the default desktop of TrueOS BSD, but has been ported to various other BSDs and Linux distributions. Despite having a website of its own, the desktop offers little information to help you get started. The project engages with the community using the Lumina Desktop Telegram channel. Besides these, there are no avenues for seeking help, so If you're stuck it's best to take the issues to your distribution's forums.

Moksha is the official desktop of Bodhi Linux but has been ported to other distributions as well. The desktop doesn't have a support infrastructure of its own and shares the one from the larger Bodhi Linux project. By contrast, LXQt boasts ample documentation and avenues for engaging with the user community should you need assistance. There are mailing lists for both users and developers, as well as fairly active forum boards. There's a wiki which covers installation of the binary packages for various distributions as well as instructions for compiling the desktop from source.

Similarly, given Openbox's age the project has extensive documentation and support from both official and unofficial sources. The project's website is a wiki with loads of information to help orientate new users. If you're stuck, there's a high probability that your distribution has an official support channel.

VERDICT

EQUINOX DESKTOP	4/10	MOKSHA DESKTOP	4/10
LUMINA	4/10	OPENBOX	8/10
LXQT	8/10		

If you're using a lightweight desktop on your regular distribution, go with the one with active support options.

LXQt 8/10

This desktop environment is a combination of the GTK-based lightweight desktop LXDE and Razor-Qt, which was an equally lightweight, but far less mature, desktop that used the Qt toolkit.

Thanks to this combination, LXQt manages to pull off the look and feel of a modern desktop without being a drain on resources. You can find LXQt in the repositories of virtually all distributions, and if your favourite distribution has a version for under-powered machines, chances are it's running LXQt.

It adheres to the old but familiar desktop metaphor, with a status bar laden with icons at the bottom of the screen. The applications menu features the traditional categorised list of apps as well as a search box to help launch apps. LXQt offers a decent number of tweakable options that help customise the most commonly used aspects of the desktop.

Moksha Desktop 7/10

This is a fork of the Enlightenment 17 desktop and features Bodhi Linux-specific changes, which the developers have been patching into E17's code over the years.

Moksha is fairly intuitive and easy to customise. It employs tons of visually appealing gadgets to display system information. These gadgets can be placed on the desktop as well as the taskbar, or 'Shelf' in Moksha speak. The desktop can be completely controlled via keyboard and you can define your own keybindings. You can also use the settings panel to influence the look and feel of your installation.

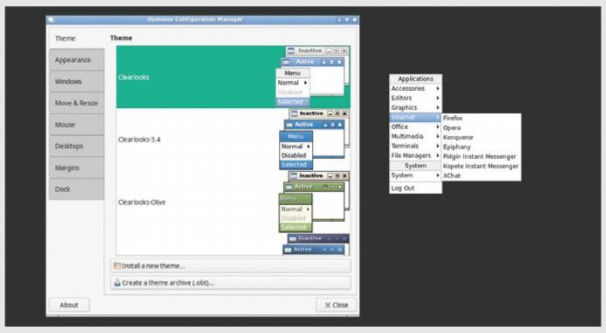
However, unlike E17, Moksha isn't readily available in the repositories of many distributions. Another issue with the desktop is its applications menu. While it does give you access to other areas of the installation besides applications, it doesn't feature a search box which can be used to quickly launch applications.

Openbox 7/10

Openbox is readily available in the software repositories of most distributions. Its minimalist yet customisable nature make it a robust choice for experienced users.

Its window manager is in fact so barebones that you wouldn't even notice it's there. All you get, by default, is a wallpaper-less background and a cursor. An application menu only appears in the right-click context menu. You can use the menu to launch apps, which run within windows with the usual controls and behave as you'd expect in any desktop.

The window manager is often combined with various other lightweight elements on the desktop. For instance you can use either *Tint2*, *Plank* or *Docky* to create a taskbar, and *PCManFM* as the file and desktop manager. You can also customise different aspects of the window manager using the *obconf* tool. Extensive manual configuration may not appeal to all users.



Performance

Do they actually perform better than the fully fledged defaults?

It's very difficult to measure performance in terms of absolute numbers due to a lot of factors. Since we can't separate the desktop environment from the apps, we can't really measure their exact draw on resources. This means the consumption will vary from machine to machine and may go up or down depending on the number of installed apps.

Also, there is no one single point of measurement. For instance, an environment could be blazingly fast to load, but its resource consumption could be really high while it's supposedly idle because of the fact that it's prefetching or loading components in the background.

On our test machine, EDE idles at about 150MB. The figures go up when an app is launched, but quickly drop back when you close the app. On a related note, it's fairly quick at launching apps, with *LibreOffice* timed at 4.3 seconds.

Then there's Lumina, which isn't designed for Linux – and that's probably why its startup takes about three times longer than EDE. It idles at around 210MB, but app launch times are only slightly slower than EDE. Its memory consumption also mirrors EDE, with highs recorded at the initial launch of apps.

Moksha is just as lightweight as EDE but its startup times and app launches are noticeably faster. While LXQt does take slightly longer to bring up the desktop, its app launch times are on par with Moksha.

Openbox is blazingly faster than the others. The startup time for the window manager depends on the number of elements it has to load. Once it's loaded, app launches are among the fastest of the lot.

VERDICT

EQUINOX DESKTOP	8/10	MOKSHA DESKTOP	9/10
LUMINA	8/10	OPENBOX	9/10
LXQT	8/10		

There's a reason why the desktops featured in this Roundup are considered lightweight, and it's not because of the disk space they require.

Plug-ins and extensions

Going above and beyond.

Just because these desktops are lightweight doesn't mean they can't provide the bells and whistles you get with mainstream desktops. As these elements inevitably increase the memory footprint of the desktop, some of the desktops bundle them inside a separate package, while others avoid them altogether.

EDE doesn't believe in offering any additional plug-ins and so you're stuck with the default functionality. On the other hand, while Openbox can be extended by plugging in various components such as a file manager, the window manager itself cannot take on new features via plug-ins, extensions or additional modules of any sort.

Lumina began life as a set of extensions to Fluxbox, a stacking window manager for the X Window System, and bills itself as having a plug-in-based interface design. This enables you to customise the desktop as needed simply by choosing which plug-ins to have running on the desktop and taskbar. But the process to change the extensions isn't very apparent, and doesn't always work if the desktop is installed on distributions besides TrueOS.

Then there's LXQt, which uses the concept of modules – essentially desktop-independent tools that operate as daemons for the local user on desktop specific operations. The desktop is further made up of several optional components such as the display manager and the power management module. LXQt's panel also supports plug-ins.

Moksha is a treasure trove of extensions, plugins and modules that add functionality to the window manager itself. It also has gadgets that are placed on the Shelf or the Desktop.

Moksha offers plenty of add-ons which you can plaster all over your desktop.

VERDICT

EQUINOX DESKTOP	N/A	MOKSHA DESKTOP	8/10
LUMINA	3/10	OPENBOX	1/10
LXQT	6/10		

Besides Equinox and Openbox, all these desktops enable you to further extend their functionality.

Configurability

Mould them as you like.

Since you've taken the steps to add a new desktop, it's only natural you'll want it to do your bidding by customising the way it works.

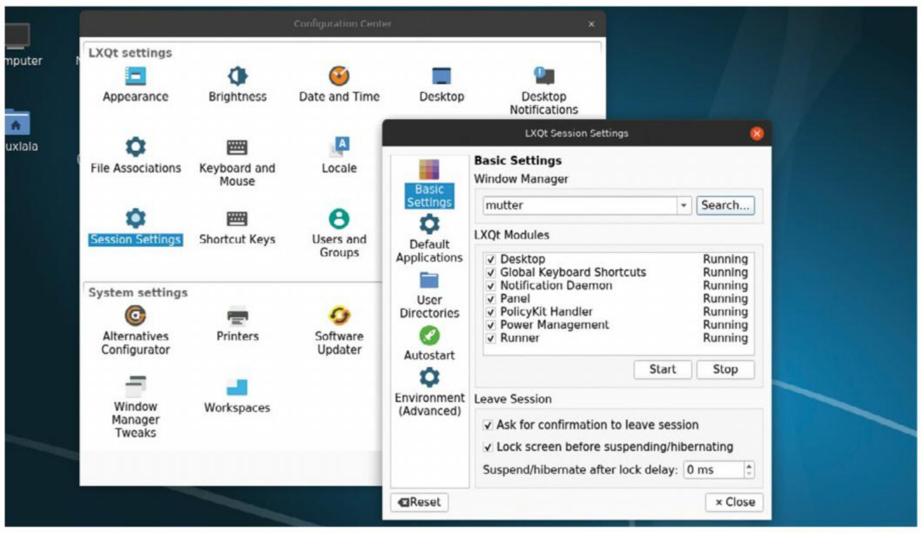
All configuration options in EDE are housed inside the configuration window. It provides six tools to customise different but very basic aspects of the desktop including wallpaper, screensaver, calendar, volume and preferred apps. Advanced users can also set various other aspects of the desktop by manually editing configuration files.

Openbox users can customise the window manager using the Openbox Configuration Manager (*obconf*) app. This helps you customise little else besides various aspects of the window manager. For instance, you can use it to change the default appearance and apply themes, and change the behaviour of the mouse and the position of the taskbar.

You can use Lumina's configuration utility to tweak various aspects of the desktop, such as the theme and window effects. You can also use it to set up default apps and define keyboard shortcuts, as well as alter the layout of the three critical aspects of the desktop – namely the desktop, panels and menu. While most of the tweaks apply system-wide, there are a couple that are restricted to the logged-in user.

LXQt's configuration centre offers similar options, albeit in a much nicer interface. But this is to be expected, as, along with Openbox, it's one of the oldest projects featured in this *Roundup*. You can use various options to configure different aspects of the desktop. While most settings apply to the logged-in user, there are several that help influence the desktop as a whole. Similarly, you can use the Moksha Settings Panel to add more bling to the desktop. You can change the theme for the desktop and even for the apps. All the elements have various other tweakable settings as well.

Besides the usual settings, there are also several advanced ones. Of note among these are the Power Management settings that help extend battery life by enabling you to adjust the time for deferring various power-intensive tasks.



LXQt's configuration manager boasts several options geared towards more advanced users.

VERDICT

EQUINOX DESKTOP	7/10	MOKSHA DESKTOP	8/10
LUMINA	8/10	OPENBOX	7/10
LXQT	8/10		

You can mould each of these desktops as per your needs.

Lightweight desktops
The Verdict

Taking app launch times as an indication, all the options in this *Roundup* are equally lightweight and responsive, and don't give us much to choose between in terms of performance. So instead we'll concentrate on their other aspects and determine our winner using the tried-and-tested mechanism of elimination.

Lumina loses out for the simple reason that it wasn't designed for Linux. It also isn't available in the repos of many distributions, which makes installation a tall order for most users.

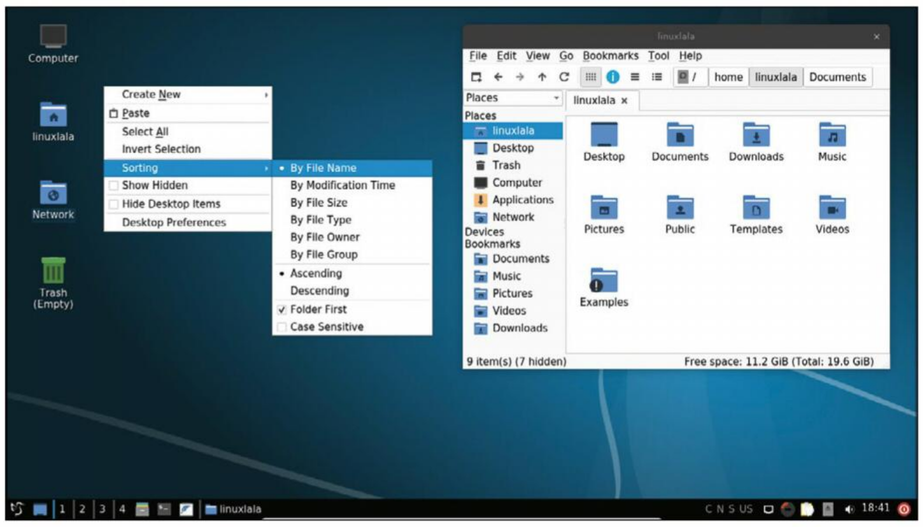
We are really pained to deny Equinox a podium finish, especially since it performs extremely well and doesn't score too badly in terms of usability. However, the project isn't as actively developed as its peers, and you can't always find it in your favourite disto.

Moksha gets on the podium for its beautiful Enlightenment pedigree. It not only looks good but is very responsive and light on resources. As with most desktops designed for a specific distribution, though, Moksha doesn't always sit pretty across other distributions.

Both LXQt and Openbox are mature projects and have been around for several years now. Openbox is surely the faster of the two, but what prevents it from claiming victory is its weak usability. Since it's a bare-bones window manager, users will have to spend some time assembling various components and writing a custom configuration file to create a usable desktop environment. While that doesn't take as much effort as it sounds, it's still an exercise and takes longer than a ready-made solution like LXQt.

LXQt is the result of the merger of the LXDE and Razor-Qt desktops. It boots very quickly and makes judicious use of available resources. Despite consuming a fraction of the resources of its mainstream rivals, LXQt looks good thanks to its Qt underpinnings, which are also the reason for its slightly higher resource use when compared to Openbox.

That said, LXQt will feel at home on a modern machine, but is still light enough to push an out-of-commission computer back into active duty.



1st LXQt 8/10

Web: **https://lxqt.org** **Licence:** **GPL, LGPL**
Version: **0.14.1**
A lightweight desktop which still manages to look good.

2nd Openbox 7/10

Web: **http://openbox.org/wiki/Main_Page** **Licence:** **GPL v2**
Version: **3.6.1**
The best option for tweakers who love spending time setting up their desktop.

3rd Moksha Desktop 7/10

Web: **https://www.bodhilinux.com/moksha-desktop**
Licence: **BSD Licence** **Version:** **0.3.0**
A wonderful option for users who love Enlightenment's bling.

4th Equinox Desktop Environment 6/10

Web: **https://edeproject.org** **Licence:** **GPL, LGPL**
Version: **2.1**
A good bet for anyone who can find it in the repositories of their distribution.

5th Lumina 6/10

Web: **https://lumina-desktop.org** **Licence:** **BSD 2.0**
Version: **1.4.0**
Optimised for BSD, it doesn't adhere to the sensibilities of a Linux distro.

» ALSO CONSIDER...

Unlike some of other *Roundups*, there's no dearth of lightweight options besides the ones featured here. There's the Sugar desktop environment that is developed as an educational desktop for the OLPC project. Although not a regular desktop environment, it's a fantastic option if you are looking for an environment custom-built for an educational computer.

One of the most popular options for cutting the flab from desktop environments is to simply run their lightweight window managers instead. We've covered some in the *Roundup* and there are plenty of others as well. There are stacking window managers like Fluxbox, Flwm, FVWM, IceWM and JWM that will all run without stressing the available resources of your computer.

Meanwhile, command-line warriors will appreciate the hackability of tiling window managers such as Qtile, i3 and xmonad. Many of them are available in the repositories of popular desktop distributions. However, they score poorly in terms of usability and for that reason aren't everybody's cup of tea. LXF

STORAGE PERFECTION

Your data is precious, so let **Jonni Bidwell** help you stash that digital cache quickly and safely.

The storage unit rental business is booming because people love to hoard things. Family heirlooms, random eBay purchases, quaint curios from happier times, 252 issues of *Linux Format* – sooner or later these things can no longer be contained in our attics and our cupboards under the stairs. It's the same with digital storage: unsorted high resolution photos and movies very rapidly fill up our hard drives, and we have to invest in more storage. Whether you want to set up your own bespoke NAS box, or just add a couple of sizeable hard drives into your tower, Linux has you covered. Through the magic of RAID, LVM, mdadm and other acronyms and abbreviations you can create fast, flexible, resilient storage for your precious bits. Data isn't indelible – just as paper yellows and turns mildewy, so bits can be flipped (a phenomenon known as bitrot), slowly, silently corrupting your files. Fortunately, next generation filesystems like Btrfs and ZFS have built in checksumming that can automatically correct for this, and layers can easily be added for other filesystems if you need them.

Building your own NAS and setting up everything by hand, perhaps by starting with a clean Debian install, is one option. It's a great way to learn how all the pieces fit together, and tune everything exactly how you need. You can build anything from a Pi connected to a couple of USB hard drives, to a case with half a dozen 10TB drives backed with a couple of lightning-fast SSDs for cache. Another option is to go with one of the many Linux/BSD NAS distros, which enable you to set everything up using a friendly web GUI. We'll look at the awesome *OpenMediaVault*, which makes it a breeze to set up even complicated storage configurations. It's ideal for storing and streaming media.

Spinning your disk strategies

Let's examine the manifold ways in which we can store our data more securely and less precariously.

Have you ever run out of space on Linux? It can be quite annoying. At best you have to move some stuff, or (shudder) do some disk-based (*tidy your room!–Ed*) housekeeping. If your root filesystem runs out of space terrible things can happen, especially if it's on a server you're not logged into. You can't log in because the system wants to write a login record, but there's no space to do that.

Even on a desktop you are logged into, strange and unpredictable behaviour might be observed. You might be unable to logout; you might, in your haste, forcibly reboot and find that you have to boot another distro to remedy the situation, as now you've corrupted your filesystem and there's no space to repair it. This is one of the reasons that having a separate **/home** partition is recommended. It's not strictly essential, and contrary to popular belief it's easy to set this up after your install.

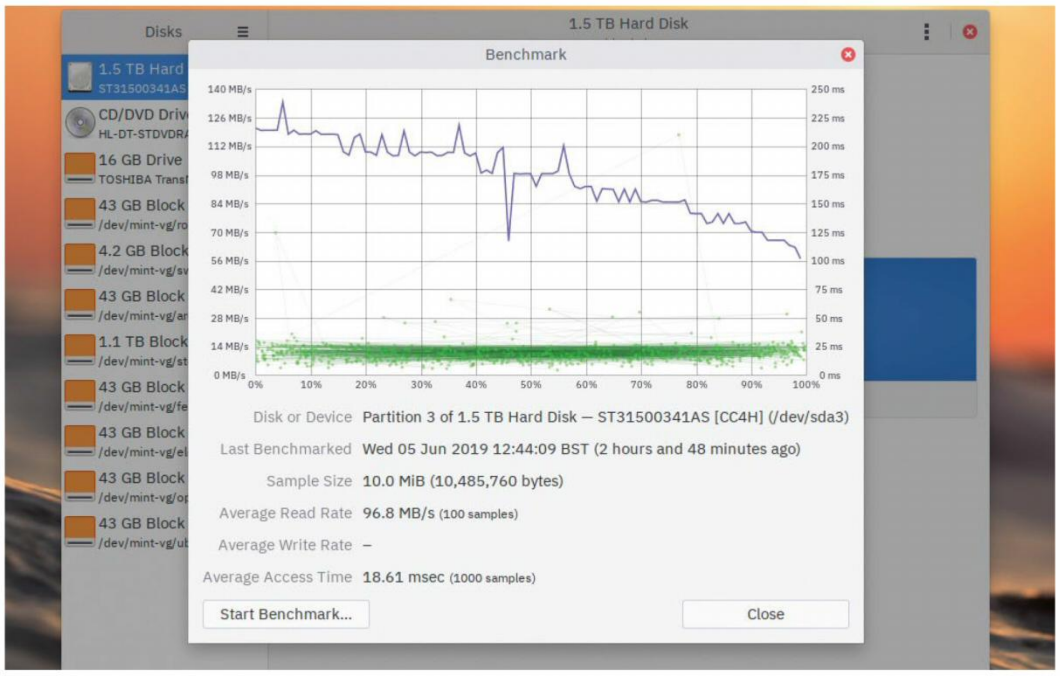
Some users prefer to store large files on a totally different device to the root file system. This is pretty much essential if you're sharing data with other users on your network, and you may even want this storage to live inside another machine. With a theoretical max of about 120MB/s, gigabit Ethernet can almost keep up with yesterday's hard drives, and consumer-level NAS devices are proving highly popular. These are great, they come in handy small form factors and many of them run Linux. They can be set up in any number of RAID (Redundant Array of Independent Disks) configurations, whether they involve mirroring, conjoining, or using parity voodoo to ensure your data stays safe even if one or more drives fail. But off-the-shelf NAS units are also often expensive or limited in how much you can meddle with them or how much extra storage can be added.

WHY BUILD YOUR OWN NAS?

"Off-the-shelf NAS units are often expensive or limited in how much you can meddle with them."

It's perfectly feasible to build your own NAS box, or indeed house a RAID array inside your desktop machine. Traditionally you'd use *mdadm* (multiple device administration) to handle the RAID stuff, and on top of that, you could use Logical Volume Manager (LVM) to abstract logical volumes away from physical partitions. Logical volumes can be resized on the fly, and may span any of the RAID devices, physical partitions and indeed drives beneath them. Nowadays LVM can handle the RAID stuff for you, though it still talks to the MD kernel layer as *mdadm* would. If you want to be hypermodern about it, you can forget about partitions altogether. Btrfs and ZFS can happily create filesystems on drives bereft of partition tables. They have their own native equivalents (volumes) as well as their own built-in RAID.

Lastly, and we say this every time, RAID is not backup. When it works, it protects against drive failure. It doesn't protect against fat-fingered deletions, and apart from the healing features of Btrfs and ZFS it doesn't protect against bitrot. So back up your files. Back them up to a RAID device, to the cloud, to a hard drive in a safe buried in the garden.



GNOME's Disk tool can run some rudimentary read, write and seek benchmarks, and draw a nice graph.

» WOE, WOE, WOE YOUR BOAT...

Linux 5.0 seems to have it in for fancy filesystems. Firstly it broke the 'ZFS on Linux' module, which flatly refused to build against it when that kernel was being tested in January. Greg Kroah Hartman has little respect for the project, stating "My tolerance for ZFS is pretty non-existent" and making clear it would be up to the ZFS team to fix it.

That they did, but the fix involved disabling some optimisations used to speed up checksum calculation (these are no longer exported by the kernel for out-of-tree module use). So ZFS doesn't perform as swiftly as one might hope on newer kernels, at least as far as checksumming goes. However, the recent ZFS on Linux 0.8 release has a truckload of exciting new features, and if you're using any non-bleeding edge distro you're probably using an older kernel.

The other kernel 5.0 snag involves *Bcache*. It appears that a kernel built with GCC 9 (as used by Fedora 30, and up until recently being tested by Arch) will lead to dodgy *Bcache* behaviour capable of "massive filesystem corruption". We don't want that, and we don't want you to encounter that, so don't use *Bcache* on such a system.

The Logical Volume Manager

LVM will remedy your capricious partition-creating desires with cold hard logic(al volumes).

We've all been there: a shiny new Linux distro comes out and you want to install it, but you already have your favourite distro installed alongside Windows 10 – and while there's free space on the drive, it's all enclosed in the OS partitions. Ideally you'd buy another drive, but that costs.

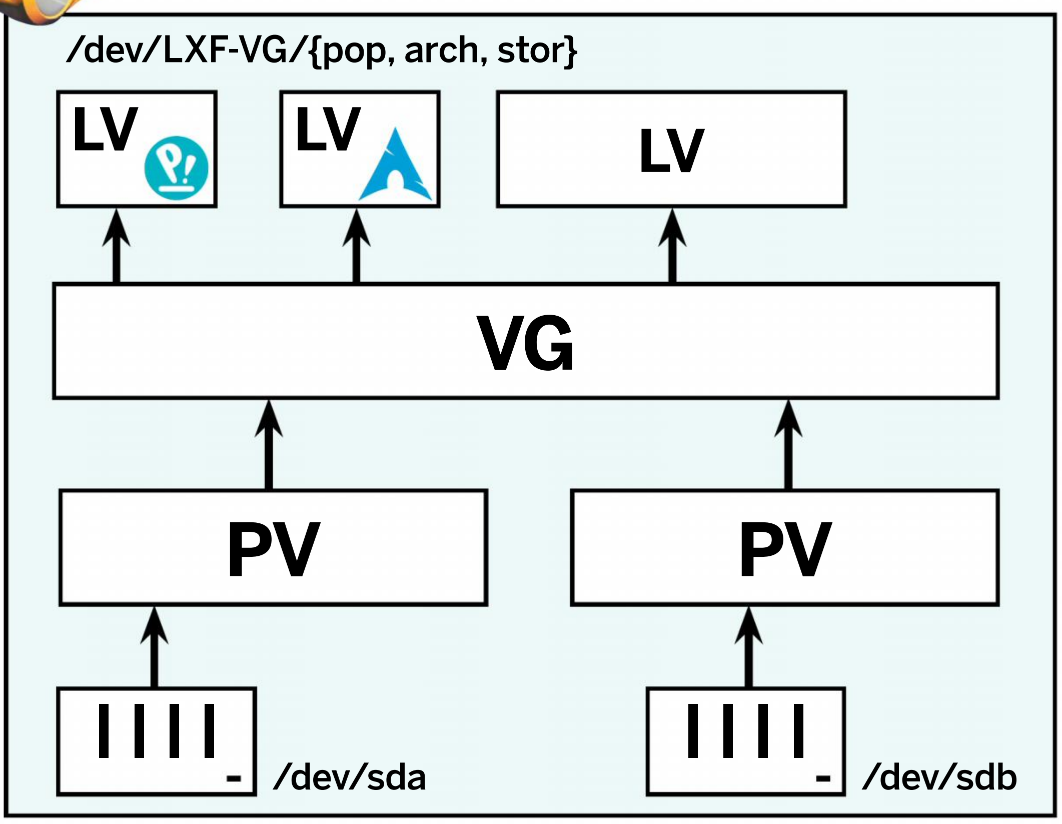
You could resize your Windows partition and move your Linux partition to the left, but experience has told you this takes time and there's a reasonable chance Windows won't boot afterwards. Wouldn't it be nice if you could effortlessly add, remove and resize partitions without extended wait times and the nail-biting prospect of failure? Well, you can't. You can't because Windows doesn't support LVM. But if all you have are Linux partitions, then good news: LVM can make your life easier.

OpenSUSE and Fedora actually default to setting up LVM on a new install, and these days most other distros will let you do this in one click. Once LVM is set up it will present Logical Volumes, which look just like regular partitions, but which may span multiple physical partitions or even multiple drives. Regular partitions, drives or other block devices (Physical Volumes or PVs in LVM parlance) can be grouped into Volume Groups (VGs, the LVM abstraction of a physical disk) and then Logical Volumes (LVs) can be created on top of those.

LVs are then totally free of physical boundaries and, unlike old MS-DOS partitioning – where you are limited to three primary partitions and if you need more you have to create logical partitions within those – you can have as many of them as you want. We don't want to oversell this, and canny readers will be pointing out not only that GUID partitioning arbitrarily allows many partitions, but that most new filesystems can be resized on the fly, albeit with varying degrees of danger. That latter point is true, but only if there's free space at the end of the partition, or unpartitioned space adjacent to it. Using LVM, we can easily add more space to a filesystem, even if that space is on another drive. Once we've added the physical volume representing the new drive or partition to the relevant volume group, there's no notion of 'adjacent' space – it's all fair game.

You can create a filesystem (anything from FAT to ZFS) on an LV exactly as you would on a regular partition, but first you'll need to set up your VG and the first step to doing that is to create the PVs on the drives that are going to house it. Let's create a simple LVM setup to see how it works. The diagram to the bottom-left shows the end result, and the wealth of acronyms.

Like Btrfs and ZFS, LVM doesn't require traditional partitions and can use entire disks for PVs. But there's nowt wrong with traditional partitions and they are necessary if part of a drive will be used for something else. There's a special LVM partition id of '8e', which you don't strictly need to use because Linux doesn't care about these things, but if you're creating LVM partitions it's not much extra effort to label them.



For the cost of just three two-letter acronyms, the Logical Volume Manager makes easy work of your storage management requirements.

Storage coming out of your ears

It's worth putting some thought into what your drives are going to be used for before making marks on the disk. By the same token it's worth remembering that you might change your mind or have different requirements further down the line. The whole point of LVM is, arguably, to make things more flexible, but certain changes will take longer than others.

Let's say we have built a new machine with two large drives, and we're fans of Pop!_OS and Arch (no, we're not going to cover installing it!), so we'd like LVs for those. We're also guilty of digital hoarding, so we'd like a large storage partition too. The first stage is to partition the disks and set them up as PVs. We mentioned that LVM can work on unpartitioned devices, but since we're dealing with a new machine we're going to need at least an EFI partition, assuming no other drives are present, and maybe a /boot partition too.

Modern versions of *GRUB* can access LVM devices, but other bootloaders (such as *Syslinux* and *systemd-boot*) cannot. Distro installers that support LVM targets generally all set up **/boot** separately, so let's do that.

Partitioning can be done from any old live system (we used an Ubuntu 19.04 disc). You can use the command line, with *fdisk* for older 'MS-DOS' style partition tables, or *gdisk* if you need partitions larger than 2TB or just prefer modern GPT partition tables; or do it graphically with a tool like *Gparted*. We want two or three partitions on one drive – the EFI partition, the optional /boot partition, and an LVM partition – and a single LVM partition on the other.

Follow the handy three-step guide below to partition your drive(s) with *Gparted*. You may prefer to use *gdisk* if you're comfortable with all things command-line. Note that if you already have an operating system set up you won't need to do steps 1 and 2. Even with no OS installed, it may also be possible to perform those steps from an OS installer – but it may not, and we want to cover all cases here. Take extra care if you're adding/moving partitions on a drive with data on it.

Once you have your LVM partitions ready, it's time to make some PVs. Run the command:

```
$ sudo lvmdiskscan
```

to see available disks and partitions. It should show you any LVM partitions you made. The *vgcreate* command can create Physical Volumes and put them in a Volume Group in one shot:

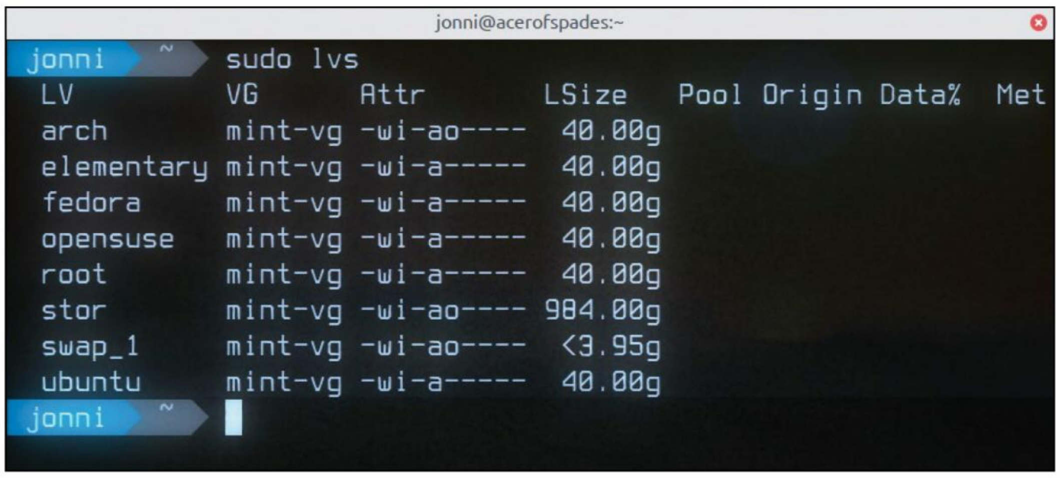
```
$ sudo vgcreate LXF-VG /dev/sdX1 /dev/sdY1
```

Change the partition (or disk) devices to the output from the *lvmdiskscan* command accordingly – you really don't want to get this wrong. This creates a VG called LXF-VG; you can of course name yours as you wish. Research the *vgextend* command to find out about subsequently adding devices and extending your volume group.

The Logical Volume Song

Now we come to provisioning our Logical Volumes. It's always tempting to provision all the space available, but this isn't really the best strategy, since shrinking volumes (and the filesystems thereon) is time-consuming and potentially risky. Enlarging them is much easier.

So a better strategy is to provision what you need, and leave some empty space for any exigencies or new ideas. So let's say we want 40GB each for our Pop and Arch partitions – that's a reasonable number, provided we're disciplined about not filling up our home directories (which of course we are, aren't we).



Apparently our tech editor maintains six distros, each on their own logical volume. Show-off.

It's tempting to just allocate the remainder to our storage vault, but if you can spare it, leave some free. We might discover a new favourite distro tomorrow, we might decide we want separate home partitions, who knows what future Jonni will want? Anyway, let's say we want to give our storage partition 1TB, assuming our drives were sufficiently capacious as to allow that:

```
$ sudo lvcreate -L 40G LXF-VG -n pop
$ sudo lvcreate -L 40G LXF-VG -n arch
$ sudo lvcreate -L 1T LXF-VG -n stor
```

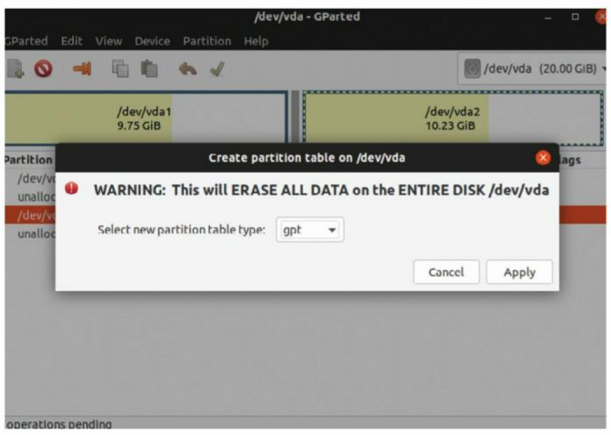
For the first two LVs, filesystems can be made during the respective OSes' installation processes; for our storage volume, we may as well do it by hand now. We'll use the ext4 filesystem; you can of course use anything you like, but if you use Btrfs beware that resizing – described later – won't be so simple:

```
$ sudo mkfs.ext4 /dev/LXF-VG/stor
```

If you need to resize your LVs – say, after adding a large drive you've just found under the editor's desk to your VG – use the *lvresize* command. On its own it just resizes the volumes, not the filesystems on top of them. However, so long as you're not using Btrfs, JFS, ZFS or anything else exotic, the `--resizefs` option can do that for you too. Note that if you shrink a LV without first shrinking the filesystem, you will almost assuredly lose data, so this option can be a life-saver.

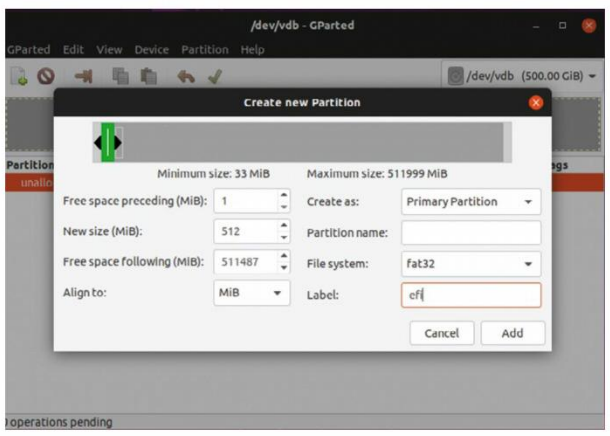
```
$ lvresize +1T --resizefs LXF-VG/stor
```

PARTITIONING FROM SCRATCH FOR LVM



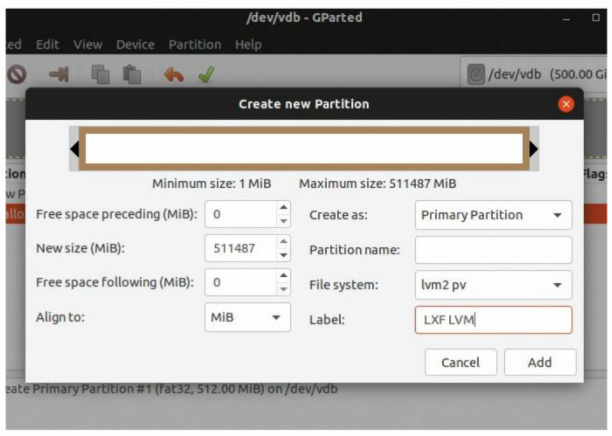
1 Create a partition table

Start the *Gparted* utility, which is included on all good live media. Select the drive you'd like to format from the top-left. From the menu, select Device > Create Partition Table. Use the GPT format unless you have some reason not to – it allows large partitions. Heed the capitalised warning before clicking Apply.



2 Create an EFI partition

UEFI booting requires one (and only one, even if multiple drives are present) FAT32 partition to house EFI images. If one exists already, do not format it! It doesn't need to be large – most distros default to 500MB. Create it by selecting the unpartitioned space and choosing Partition > New. Choose 'fat32' from the filesystem drop-down menu.



3 Create other partitions

You may want to create a separate boot partition, in which case repeat the previous step. Everything else is going to live inside our Volume Group, so we'll create a single large partition with 'lvm pv' as the filesystem. If you're going to enrol other drives into your LVM, create partition tables and PV partitions there too.

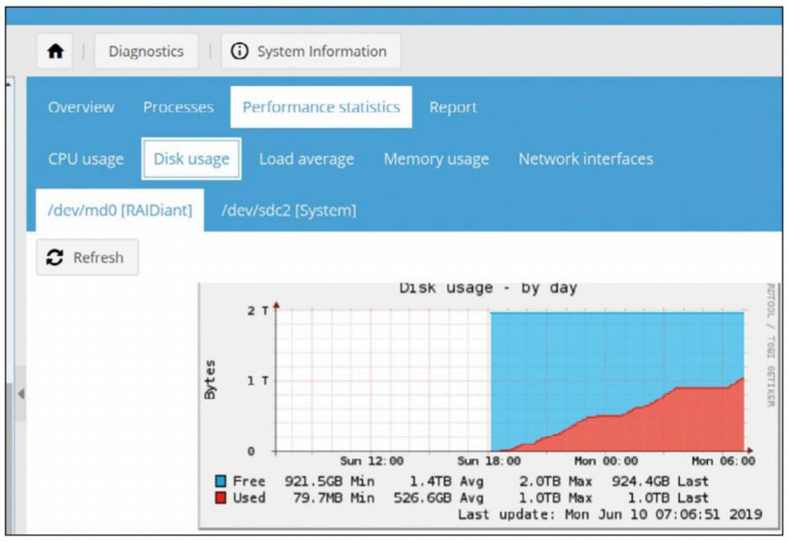
RAID with OpenMediaVault

Set up your own NAS box with all the redundancy your data deserves, and a funky web interface to boot (if you'll pardon the pun).

Hard drives fail. They wear out, they get dropped, they get fried in electrical storms. Some of them, doomed from the start, come off the manufacturing line worse for wear. Potential drive failure is a good reason to back up your important files, but it's an even better reason to deploy a RAID configuration. RAID (Redundant Array of Independent/Inexpensive Disks) enables you to use multiple drives to increase the resilience of your storage.

RAID1 is the easiest configuration to explain: it mirrors the contents of one disk to one or more others. This means that all but one of the drives involved can fail and you still won't lose any data. RAID0 doesn't have any such redundancy, but rather 'stripes' data across drives so that it can be read and written sequentially, speeding up access. We're not interested in RAID0 here as it plays fast and loose with your data.

Some RAID1 implementations will see improved read times, since reads can be parallelised across drives. These two schemes can be combined to give either RAID1+0 or RAID0+1, which give striped mirrors or mirrored stripes. But thanks to the wonders of mathematics, there are other options too. By striping data across at least two drives, then adding parity information about that data to a further drive, we can create a system where if any drive falls over, the data that was on it can be reconstructed from the contents of the others. This is known as RAID5, and we can go further: RAID6 allows for two drives to fail, by having two parity blocks (but requires at least four drives).



All kinds of glorious graphs await you in the Performance statistics section. How quickly we filled our RAID...

In the olden days, RAID used to require a dedicated I/O card, and for enterprise purposes they're probably still a good idea – along with an Uninterruptible Power Supply, backup generators and round-the-clock monitoring. But the parity calculations that would have burdened CPUs of yore are no problem nowadays, and RAID is easily implemented in software.

In Windows it's called Storage Spaces, and in the Linux kernel, it's known as MD (Multiple Device). The tool that manages it is called *mdadm*. As with LVM, *mdadm* combines drives and presents a single abstracted block device, for example **/dev/md0**, which we can put any filesystem we like on. Also like LVM, *mdadm* can operate on either partitions or entire disks. It does, however, get awfully confused if you try to use disks that have in past lives been part of MD arrays, so it's important to zero the superblock if you've done this:

```
$ sudo mdadm --misc --zero-superblock /dev/sdX
```

To create the three-drive RAID5 setup we built using a Fractal Design Node 304 case kindly supplied by the manufacturer, we would do:

```
$ sudo mdadm --create --verbose --level=5 –
metadata=1.2 --chunk=512 --raid-devices=3 /dev/md0
/dev/sdX /dev/sdY /dev/sdZ
```

It will take some time to prepare the array, although it can at this stage accept a filesystem and be mounted in 'degraded' mode. As before you'd create that filesystem with:

```
$ sudo mkfs.ext4 /dev/md0
```

You can monitor progress of the array build with:

```
$ cat /proc/mdstat
```

Alternatively, follow our majestic six-step guide to *OpenMediaVault*, on the opposite page.

» GET SMART

Having a huge amount of storage capacity at your disposal is all well and good, but besides chucking data upon your platters (or SSD cells), it's worth devoting some time to checking their health. You can perform a non-destructive read test at any time, and modern drives all support SMART attributes, which if monitored dutifully can warn you of imminent drive failure.

The *smartctl* utility (part of the *smartmontools* package) lets you run a number of different tests. The easiest is a short test:

```
$ sudo smartctl -t short /dev/sdX
```

You'll be given a guesstimate of how long it takes; as the name suggests it won't be more than a couple of minutes. After that time you can check the results with:

```
$ sudo smartctl -l selftest /dev/sdX
```

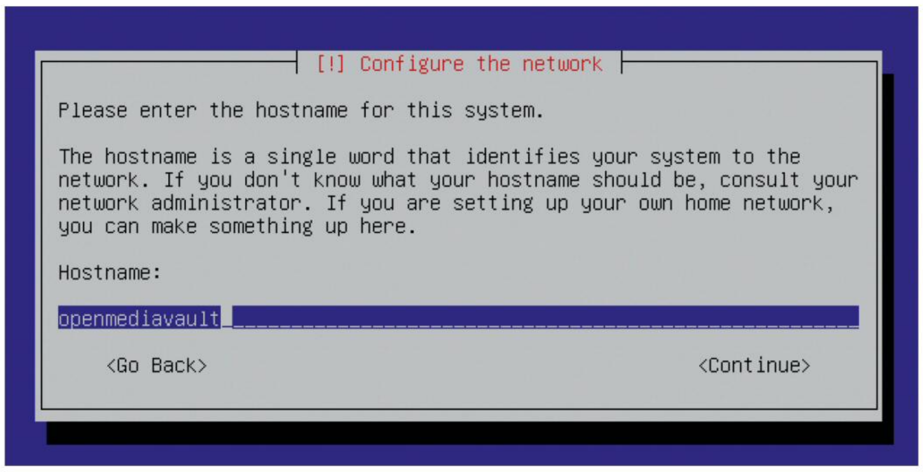
Replace `short` with `long` to carry out an extended test in the background. This estimated five hours for a 1.5TB drive, but we wanted to go home before this finished, so we were forced to curtail the test with:

```
$ sudo smartctl -X /dev/sdX
```

OpenMediaVault can configure SMART parameters from the Storage section in the settings menu. This enables you to schedule scans and receive alerts, for example regarding high temperatures. Activate it per drive from the Devices tab. SMART will passively detect bad sectors and avoid writing to them.

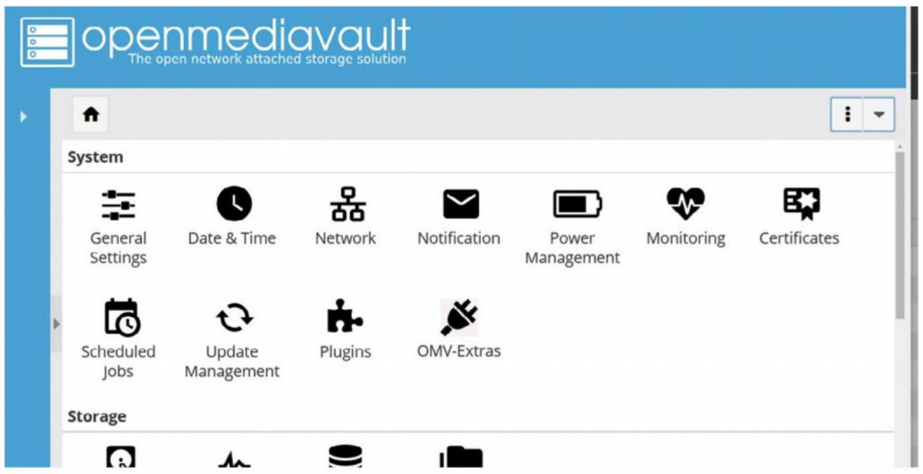
But if you're serious about storage security, it's worth looking into the *Badblocks* utility, which painstakingly tests each sector's integrity. It also enables you to do a destructive read-write test, which for obvious reasons you should do before you start storing data.

INSTALLING OPENMEDIAVAULT



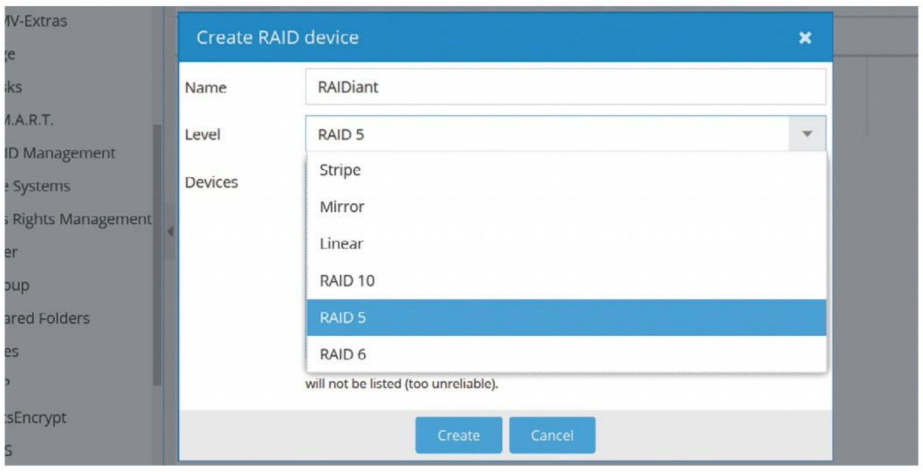
1 Installing

Write the ISO from the **LXFDVD** (it's in the **OMV/** directory) to a USB stick using *dd* or *Etcher*. Boot that on your soon-to-be NAS box. The interface is pretty old-school, but don't be intimidated. Just work your way through the options and hit Go. Be aware that the installation will indiscriminately destroy everything on the target drive.



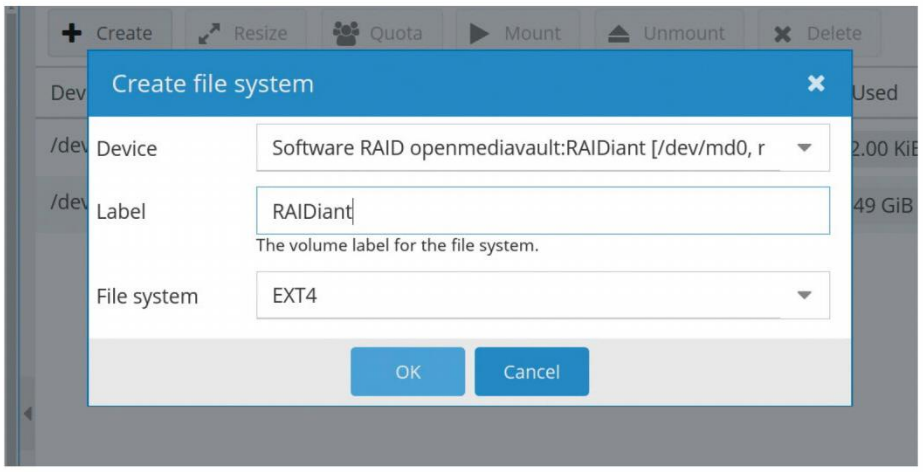
2 Log in

Reboot and the machine should tell you its IP address and offer a login prompt. It may instead helpfully tell you it can't see the network, in which case log in as `admin` with the password `openmediavault`. Run `ip a` and hopefully you should see an IP address. Visit this address from a web browser on your network.



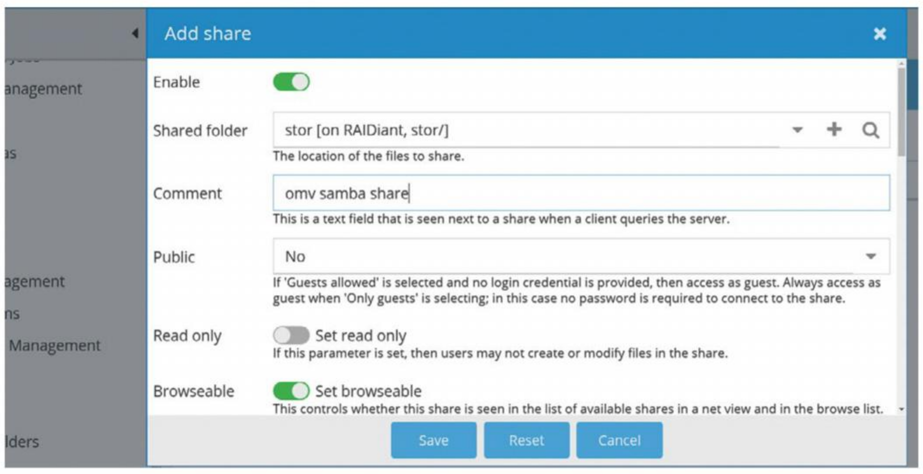
3 Create RAID

Log in to the web GUI with the previous credentials. Go to Storage > Disks and select all the drives you want to include in your RAID and select Wipe–you have installed at least three hard drives, right? Use the quick option to save time. Now go to RAID Management and select Create. Choose your RAID level and check the drives you want to involve.



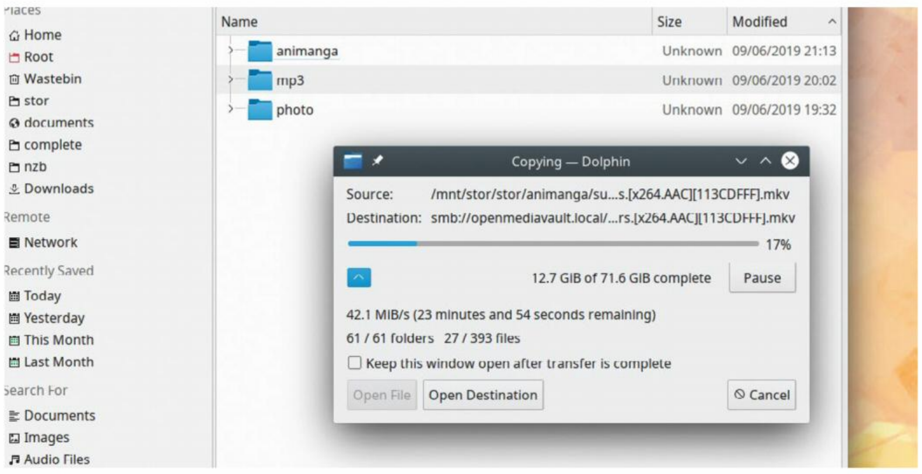
4 Create fs and share

Really you should wait till *mdadm* has synced the RAID before you do this, but it's probably fine. Go to File Systems, and click Create. Select the **/dev/md0** device and choose a filesystem. In order to use this we need to assign it a path. Go to Shared Folders, choose a name, select the RAID device and choose a path.



5 Enable CIFS share

We need some way to interact with our RAID array. We can do that via SSH (configured by default) or with Samba, which also lets other OSes play along and in our testing was faster. Go to Services > SMB/CIFS and enable the service. Then go to the Shares tab, select Add and configure your share and its visibility.



6 Fill with data

Most file managers will let you browse Samba shares by pressing Ctrl+L and entering **smb://** followed by your NAS's IP address. You should create a user (maybe more) on OMV and log in as that user when prompted. Then you can start to fill your NAS; we managed to get about 50MB/s over gigabit Ethernet.

Next generation filesystems

Level-up your filesystems and plug your OpenMediaVault install into, er, some functionality-enhancing plug-ins.

Filesystems and indeed the media that they live on have come a long way. Cassettes, cartridges and punchcards didn't even have a filesystem to speak of: data was just read as a single stream. From there we moved on to directory-less layouts, through to 8.3 filenames, then on to more recognisable features such as permissions and, latterly, journalling. Traditionally, storage has been stratified such that the RAID layer – be it software RAID via MD in the Linux kernel, hardware RAID via a dedicated controller, or so called 'fake raid' through the BIOS – is totally separate from the filesystem layer. As we've seen, though, RAID only protects against drive failure.

» GENERATION Z

ZFS was released for the OpenSolaris project in 2005, so it predates Btrfs, the 'other' next-generation filesystem. When Oracle acquired Sun in 2010, no further OpenSolaris code was released, leaving ZFS languishing. Even before this though, the CDDL licence under which OpenSolaris was released had been considered incompatible with the GPL, so no ZFS code ever made it into the Linux kernel.

The OpenZFS project has continued to work on ZFS and provide CDDL-licensed code for BSD, UNIX, Linux and macOS. Under the aegis of OpenZFS, the ZFS-on-Linux project provides an out-of-tree module for Linux, which Canonical's lawyers deemed fit to include in the Ubuntu 16.04 repositories.

It's not widely used, but this may change in Ubuntu 19.10, which is currently exploring enabling this module by default, and hence offering the option of having a ZFS root filesystem. Of course, this is possible just now; it's just a little convoluted since you have to manually prepare the live environment, pool layouts and datasets. See all the gory details for yourself at **http://bit.ly/lxf252zfs**.

ZFS is supported on OpenMediaVault through a plug-in from the OMV-extras repository, which will save you some command-line incantations. One of the oft-mentioned but rarely understood criticisms of ZFS is its memory requirements. It relies on a lot of data staying in RAM, and that data's integrity is paramount. As as a result it's recommended to use only ECC memory with ZFS. Only if you use deduplication does the rule of thumb of the order '8GB RAM plus 1GB per TB of used storage' apply.

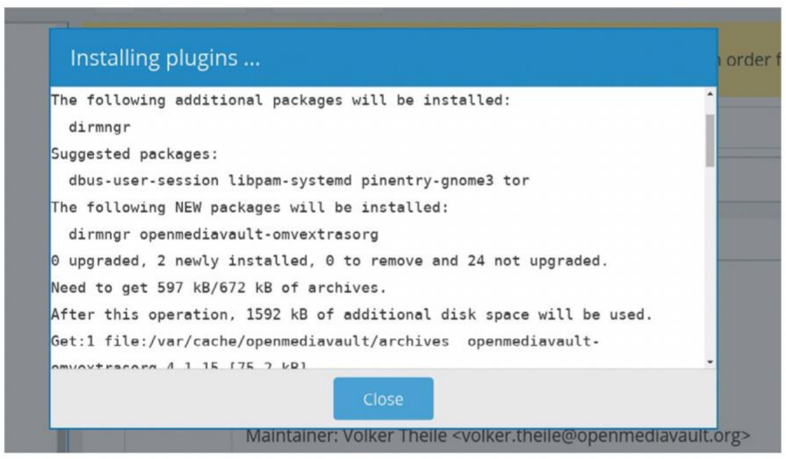
If you have, say, a two-drive RAID1 system, and one of your drives starts writing jibberish without outright failing, then the RAID layer can deduce something is wrong (since the same file will have different contents) – but it has no way of telling *which* drive is broken. There's no easy way around this that doesn't involve either adding a new checksumming layer, or combining filesystem, redundancy and checksumming into one. Unsurprisingly, it's this latter approach that has gained popularity, and it's one of the defining features of Sun's ZFS *(see box, left)*, Microsoft's barely seen ReFS, and Linux's Btrfs. Apple's new APFS filesystem includes checksums for metadata, but not user data.

We'll concentrate on Btrfs here, because it's been a bit neglected in the preceding pages. Besides the extra resilience it provides through using 'copy on write' – in a sense, the natural evolution of a filesystem journal – it also has its own built-in device-spanning capabilities. This means there's no need for *mdadm* or LVM, although you can use Btrfs on top of these if you want.

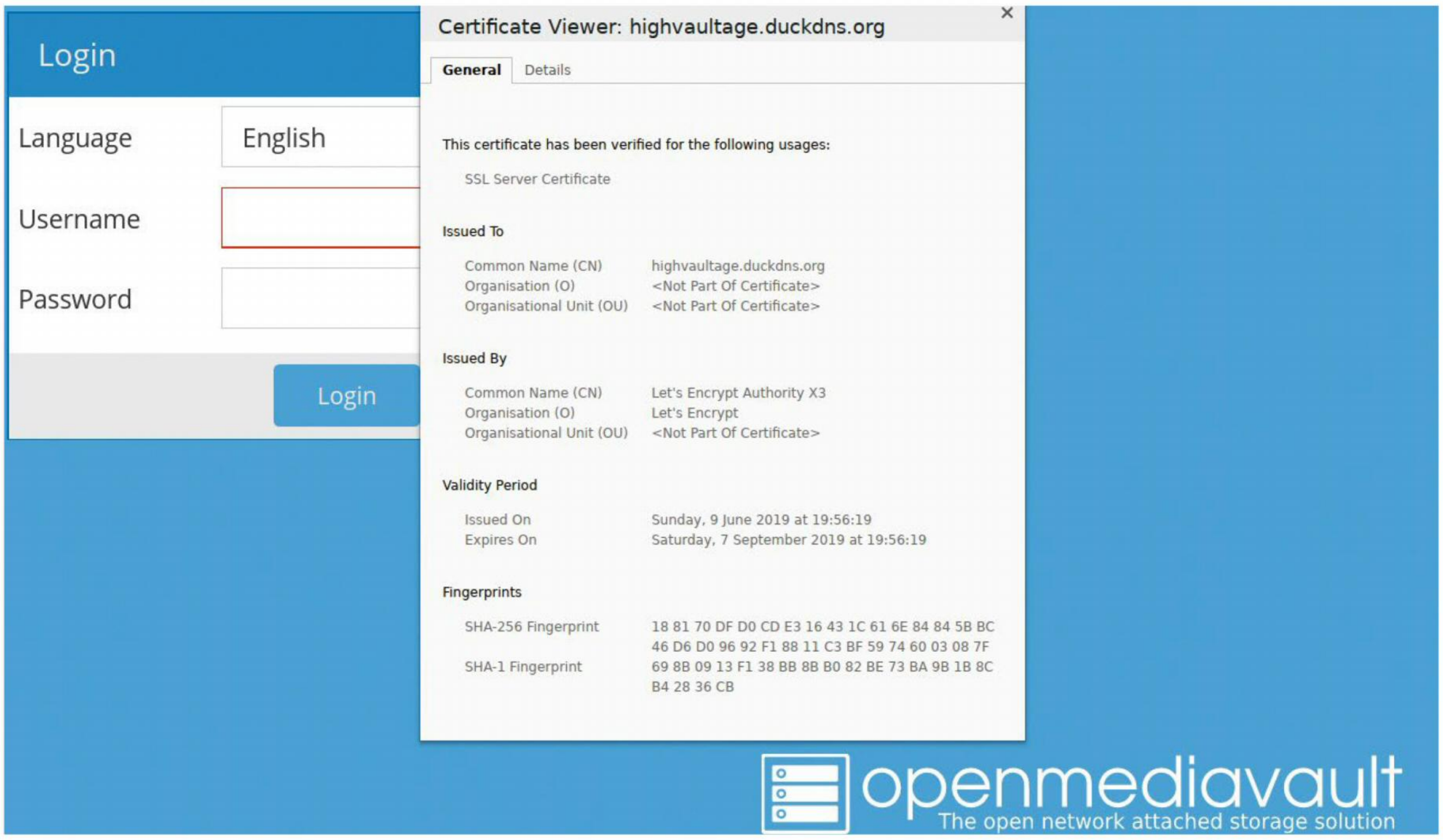
Another great feature is snapshots. You can take a freeze-frame of a whole filesystem and it will initially take up zero space. As that filesystem changes, the changes are stored incrementally in the snapshot and it begins to grow. This means that with very little effort you can take daily snapshots of your drive and revert any one of them when something goes wrong, be it fat-fingered deletion, cosmic-ray strike or SATA cables becoming the new catnip. Btrfs snapshots are one of the ways that Linux Mint's Timeshift backup tool can do its magic.

Next gen-ing your NAS

Using SMB to access your OpenMediaVault shares is all well and good, but you'll see faster transfers if you use *rsync* to copy files instead, which you can also configure to push-to and pull-from remote *rsync* servers within the Services section of the GUI. Neither of these protocols are very useful for streaming movies, and indeed lots of movie players baulk when given remote URIs (such as **smb://**) instead of local paths. Especially, as it turns out, on KDE Plasma: either use



OMV plugins are shipped as .deb files, and you can enjoy watching Apt install them just like on Debian



You can get a Let's Encrypt certificate for a free duckdns.org subdomain and access your NAS securely from anywhere.

VLC or mount your shares as CIFS volumes from **/etc/fstab**. This will save you getting lost in a clueless web of KIO/Phonon/Gstreamer unholy couplings here.

It's possible, but not necessarily smart or fun, to mount these shares remotely, provided you're willing to open up the right ports (137, 139, 445) on your router's and possibly your distro's firewall. We'd recommend sticking with SSH mounts for remote access, which just involves TCP port 22. When you create a new OMV user (as we recommended in step 6 of the walkthough), you can add them to the SSH group to enable them to access. By default, luxuries like the *Bash* shell and a home directory aren't set up, but they can be in a few clicks if you want to work on your OMV box like a regular Debian server.

You can also add your user to the 'sudo' group, but for remote access we'd strongly recommend setting up public key logins and disabling passwords. You can add a public key by selecting your user and choosing Edit > Public keys > Add. If you've already set up a key with *ssh-keygen* on your local machine, you can convert it to the required RFC4716 format with

```
$ ssh-keygen -e -f ~/.ssh/id_rsa.pub
```

and paste the output into the box.

Installing the OMV-Extras.org plug-in will avail you of a whole bunch of community plugins. These include Let's Encrypt integration, so that remote connections can access the admin panel securely over HTTPS. It also allows you to set up *Docker*, which can in turn run Nextcloud and plenty more. OMV-Extras also enables you to enable the *Plex* repository, which would let your NAS stream movies to anywhere with a web browser. The GUI-based Let's Encrypt setup requires you to (temporarily) forward port 80, which might be tricky on some routers since they'll want to serve their own web interface there – but it's absolutely a wise thing to do if you're accessing your vault from far away.

One last thing: you might suddenly find you can't log in via SSH – in our case it was after changing the default password, which is another thing you should definitely do – either using the admin account or via public key with any added accounts. This could be a strange but recurrent filesystem permissions problem. You'll see an error like this in the logs:

```
sshd[21546]: Authentication
refused: bad ownership or
modes for directory /
```

If you can get a root shell (add your user to the sudo group from the GUI), you can fix this with:

```
$ sudo chmod og-wt /
```

As usual, we've learned things the hard way so that you don't have to... LXF

Small but perfectly formed, the Fractal Node 304 will soon be filled with drives.

» HARDWARE CONSIDERATIONS

We've already mentioned that our new best friend Shawn from **www.fractal-design.com** lent us one of its Node 304 (about £80) cases. It's an ITX box that can fit six 3.5-inch drives in it, but is no bigger than two shoeboxes, so is ideal for hiding under the stairs. It's a little tricky to work with if you have a large heatsink, but it has excellent airflow with two fans at the front and a large fan at the back.

It's always tempting to opt for cheaper components, or even second-hand ones – but consider how much your data is worth. Consumer-grade hard drives are not built for the kind of longevity one expects from a NAS box. It's also worth noting that RAID5, and even RAID6, isn't really considered viable for critical data on large drives. This is because in the event of an error, so much data is read off the other drives that the odds of a read error occurring and the rebuild failing become significant.

Of course, if you're just playing then a Raspberry Pi connected to a powered HD caddy via USB will let you see how all this works. A slight improvement would be to go for something like an ODROID XU4, which has onboard SATA. If you're not using ZFS or planning any crazy antics with the *Docker* plug-in, then 4GB of RAM and a modest CPU will be fine for running *OpenMediaVault*. Large-capacity SSDs are becoming cheap and these can be used as a cache for your spinning rust drives. The required weapon is known as *Bcache*, and it's a shame we can't talk more about it in this tiny box.

www.fractal-design.com/home/product/cases/node-series/node-304-black

PICTURE IN A MAINFRAME

Jonni Bidwell wants to know if he could install Linux on the mainframe at Future Towers. The Open Mainframe Project's **John Mertic** has all the answers.

John Mertic is Director of Program Management for the Linux Foundation's Open Mainframe Project (www.openmainframeproject.org), an effort to bring open source knowledge, and of course Linux, to mainframe computing. You may be forgiven for thinking mainframe machines were all clunky relics running obscure COBOL code, but the modern mainframe is a thing of technical beauty. While the cloud offers elasticity, portability and spares you the woes of managing your own infrastructure, mainframe(s) offer resilience, security and performance.

The project uses Zowe, the first open source software to run on IBM's z/OS – and yes, you can now run Ubuntu on your mainframe. Besides being a mainframe man, John is also Program Manager for a number of other Linux Foundation projects, including: Open Data Platform (**www.odpi.org**), a non-profit dedicated to standardising the big data ecosystem; the R Consortium; and the Academy Software Foundation (**www.aswf.io**), whose goal is to further the role of open source software in creative industries. He was good enough to spend some time chatting to us at the Linux Foundation's Open Source Summit back in October 2018.

Linux Format: **Mainframes always seem inaccessible to the lay person. They're bulky and expensive and it seems like they can only be operated by people with crazy hair (*shouldn't throw stones?–Ed*) in white coats. Is this changing?**

John Mertic: A couple of years ago there was a student who bought a mainframe off eBay. He had it in his parents' basement[1] and he figured out how to get all the software and everything running on it. The mainframe community was blown away – he got to bring it to conferences and everything. I think IBM helped give him a bunch of missing pieces that he needed; I think he works for IBM now.

I guess if we classically look at where you'd see mainframes in a historical context, my kids would tell me it's the spaceship earth ride at Epcot [at Walt Disney World in Florida]. You're riding by and you see the mainframe setup and the dudes with big hair. It's kind of a shame – if you search Google for 'mainframe' you see so much of that being the context. I think the mainframe community's quite sensitive over that, but I'm a big fan of how we look at the legacy of these things.

When the current generation of mainframes rolled off the line in 1964 they were all designed with security, performance, scalability and availability as the core tenets. And every design decision and every hardware addition builds on those. That's what's kept that ecosystem rolling, even up to the boxes today. The latest one actually fits into a standard rackmount unit; before, they used to be a lot bigger. It's interesting technology, and if you're running SUSE or Ubuntu on it, you really can't tell the difference between it and a regular server – it all just works the same.


John Mertic does not have crazy hair.

LXF: **There's a lot of talk at this conference, most of which is very confusing to me, about new technologies like IoT, microservices, fog computing, edge computing. But I think it's easy to forget we owe a lot to mainframes, and they're still a big part of computing today. I was talking to the IBM folks last year and they told me that every card transaction and every flight booking goes through a mainframe computer. So even if we don't see them they're still very much there, and very much aren't going anywhere soon, despite often being seen as a relic. What is the Open Mainframe Project all about, and how will it help with this retrospective?**

ON MAINFRAME USERS

"You have a community that's always been very much open source and collaborative-minded."

JM: The Open Mainframe Project is a foundation launched to support the growth in open source projects on mainframe. Very astute, by the way, your knowledge of the mainframe ecosystem, and where we've come from. It's really interesting in an open source context. If you want to trace back the roots of open source back to predecessor organisations, you'll find it all traces back to 1955 and SHARE. SHARE is a mainframe users' group; it still runs today, but it was the concept that all these mainframe users, programmers, administrators, systems people would all get together and share source code. Back then they were using things like microfiche and tape and things like that, but it was that same idea of sharing (that we have today).

So you have a community that's always been very much open source and collaborative-minded, but never used that terminology. That's always been a root part of this ecosystem. The open source movement came alongside of this. So this project is about how can we pull all of those different collaboration efforts together, so there's a natural home where you would go to get support. Whether you're an open source project and wanting to support the mainframe, or if you're already on the mainframe and you want to support open source, that's what we're here to do.

LXF: **Which industry partners are you involved with?**

JM: Lots of organisations – obviously IBM, they're the primary mainframe vendor, I think they have something like 90 per cent of the market, so that's the hardware we focus on predominantly. But we also

1) www.ibm.com/thought-leadership/passion-projects/careers-mainframe/

have Linux vendors such as Canonical and SUSE involved. Then there's application vendors like CA, Rocket Software and Docker. We also have universities and IBM systems partners involved.

There are also just general end-users of mainframe in the community. I don't know if ADP is well-known in Europe, but it's the largest payroll processor in the US. Most Americans and Canadians get a pay cheque or a W2 from ADP, and it's all done from Linux on a mainframe. It's a pretty wide spectrum, it's also a wide spectrum of generations. So you have 70-year-old systems programmers, you have middle career people, you have new developers. You have all these intermixed dimensions that are coming together.

LXF: Banks in the UK are rubbish. I mean, they're probably rubbish everywhere but in the past few years we've had some pretty major outages: people's pay not being credited, people not being able to access funds, mortgage payments bouncing. And a lot of the time this is blamed on failed mainframe updates. The feeling I get is that we're losing expertise here, as people that understand these things are retiring and, well, expiring. I mean, who learns COBOL nowadays? Can the Open Mainframe Project ensure this ancient knowledge is taught to a new generation, so that our funds are not at the mercy of fat-fingered updates?

JM: There is a generational gap, and I think that's one thing the mainframe industry is forward-thinking about. It's a 54-plus-year-old industry, how's it going to get through the next 54 years? It's going to be through getting that next generation involved. There's a number of different efforts that are underway. I think one piece is that the drive of Linux and open source and mainframes already having that commonality between it.

Frankly speaking, mainframe can't be replaced and not just because of the applications that are already locked into it, more because of what it can provide. If you're an organisation and you need an application that needs the dials of security, performance and reliability all turned up to 11, then no cloud system can do that for you, no distributed system can do that for you, only the mainframe can do it. That's why these organisations are centred on the fastest commercial processors on the planet running inside mainframes.

Being able to have encryption as your data is read into memory and moves back onto the disk – hardware-based encryption – that's only on mainframe. All of these innovations happen here, resource-intensive, high transaction computing happens here. It's not going away and it's not going away because there's nothing else that can replace it.

So the big question is how do we get that next generation engaged, but then also how can all of these applications that are built on mainframe, that are perfect for it, that are completely designed for it, how can the data generated from them be taken advantage of in the rest of the organisation? Because, and I'm sure the UK probably sees the same thing, the mainframe group is always silo-ed off away from the rest of IT, and that becomes a huge problem.

So on one hand I think the emergence of Linux on the platform, the growth of Linux on the platform is a huge piece there. We're already seeing projects, we've worked with Kubernetes, Cloud Foundry, Openstack, they're all starting to pull together. Organisations can, from a devops perspective, manage all of that infrastructure in unison, versus having specialised management. The other project that we've been working on, that we launched in summer 2018, is a project called Zowe (**www.openmainframeproject.org/projects/zowe**) and what that's focused on is the z/OS side of mainframes. That's sort of the legacy operating system there.

Before Zowe, the only way you'd interact with it was over what they called 3270 terminals, which in layman's terms are green screens. What that does is it puts a set of REST APIs on top of that, so that any modern application can easily tie

» WORKING ON THE BLOCKCHAIN GANG

LXF: It's hard for me to hear 'blockchain' without thinking about cryptocurrencies. At present Bitcoin can handle about six transactions per second, Ethereum (in its current form) can do slightly more, but that's nothing compared to Visa, which handles thousands every second. Do you see those blockchains evolving so that they could handle a similarly high transaction rate?

JM: I wouldn't call myself anywhere within the ballpark of an authority on finance, but I think the most interesting use cases for blockchain are not going to have anything to do with finance. I've been around enough to know how new technologies work. At first it seems like it can solve every problem, then you realise there's some problems it can't solve and interest wanes, and then you find there's some things that it is really good at and there's a resurgence.

I feel like blockchain is just at that point now. I think logistics is going to be one area where it's going to be huge. The ability to be able to coordinate down your supply chain is something that most companies do in an ad hoc manner today. If they could have a trusted way of managing that, boy, that will blow a lot of things away. There's just so many problems that would get solved. Logistics and healthcare stick out to me. With healthcare, you want to share your records securely with select people. That's an interesting problem. There's also been a couple of good use-cases in government – having a decentralised system for land management, for example. Certainly in the States, and probably here too, you'd traditionally have to go to a county planning office and fight with bureaucracy to get the required permit. Each locale doesn't necessarily have the greatest way of dealing with that process.

We all got excited over cryptocurrencies, and there's obviously something to that. But it's going to be something like the second, third, fourth-largest use case for blockchain.

in to all the mainframe data. Additionally it adds a command line interface, just like what you have if you SSH-ed into a regular Linux box. There's a web UI portion there too, so you can basically build desktop apps in a browser that ties right into the mainframe. But it's all Electron/Node.js built, so you can easily integrate different sources as well. That project has the opportunity to revolutionise, so that (for) all these legacy z/OS apps that are out there, and are going to be there forever, at least there's a way that data can be migrated back into the rest of the infrastructure. So now that's helping to make more commonality amongst the toolsets that people are using.

So we have those two focuses, and I think the third focus that we're working on is our internship program. They actually presented here at the summit today – we brought the gang along. As what we're really trying to say is "Hey students, you're into open source, there's a lot of open source here on mainframe, there's opportunities. Come spend the summer with us and work on it". And they make great contributions – we've had students that have made contributions to Hyperledger, Kubernetes, blockchain. We had one student that ported Alpine Linux over to the platform.

Then all of those students then come out of this program, and they're already connected to mainframes, they're already familiar with the technology, they see the great opportunity for, frankly, very high-paying jobs. They're some of the best-paid new graduate Computer Science jobs out there. And (through the program) they already have a foot in the door and a career path. Then that's helping all those companies – going back to your point before – that are trying to backfill all those positions resulting from the retirees.

LXF: Yep, occasionally you see job adverts for VAX experts. Does anyone understand that any more? I mean, I once saw a VAX machine – it was being used as a coffee table. Very sturdy. Not sure if it still booted.

JM: Well, VAX is probably a whole different ball of wax there. But now there's a lot of these technologies out there that are going to be around forever and I think that's an interesting challenge to the open source community. What can open source do to unify all these? Maybe we're taking the lead here with our Project.

LXF: With Openstack and Cloud Foundry and Kubernetes and such, you can run all these at home on a bunch of virtual machines. It might not run well, but for illustrative purposes and to get a handle on them, you can do that. You can't very easily get an old mainframe and plug it into a normal power supply.

JM: The interesting thing is, by having that ability where the software layer is common across all those architectures, now you can use the architecture that best suits what you're trying to accomplish. I mean, you and I are lucky to live in an age of technology. That wouldn't be the case 20, 30 years ago – you'd get a handful of vendors, and once you purchase your hardware from them you're stuck in their ecosystem, you're not getting out. Now you have Linux as this commonality across all of these architectures, and then you have the value-adds that are popping on there, Kubernetes and all these different technologies that are all ubiquitous across all of this.

Now you can say "You know what, if I have an application here that I just need ridiculous performance on, I have high security needs, then I can flip it over and I can run it on the mainframe. What if I need to run at the edge? OK, fine, I'll push it onto an Arm box. And if I need some elasticity on the platform, then it can go in the cloud". You have all those choices. We live in an age where we have a gigantic menu of computing choices, which is fantastic. It's a great thing for organisations to take advantage of, because they're becoming more and more heterogeneous in their environments.

LXF: Apart from my two examples earlier, which I practised in the bathroom mirror this morning...

JM: Ya nailed it, buddy.

Nor does John have a white coat – or at least, he's not wearing one.

THE NEED FOR TRAINING

"There is a generational gap, and I think that's one thing the industry is forward-thinking about."

LXF: Thanks. So apart from those, I also learned a little bit about pervasive encryption. That's something that's peculiar to mainframes, it would be frightfully complicated to cloud-ify it. Can you tell our readers something about it?

JM: Pervasive encryption is an IBM technology on the latest z14 boxes. The nutshell is, as data is hitting memory, it is automatically hardware-level encrypted, not at software level. It's encrypted before it's even hitting the disk, so that's an additional layer of security to that data there. Mainframes are quite unique because mainframes have never been hacked – they're the most secure things on the planet for a reason. And pervasive encryption just continues to add to that, so if you have those sorts of requirements then that's what you need. We're seeing other technologies take advantage of this too; IBM is working on a secure containers offering that's using KATA containers. We had a student this summer leveraging that same technology, again to make this platform as secure as it needs to be.

LXF: I always like to pore over specifications and compare them to this pitiful machine from 2011 that I do all my work on. So how powerful are these machines, really?

JM: I think the z14s have 5.2GHz processors *(they do indeed – Ed)*, and they have about a hundred of them. They don't have what we would call RAM, they call it RAIM (Random Arrays of Independent Memory) – it's fault-tolerant and they can have up to 32TB. You can hot-swap any single piece of a mainframe out while it's running and it just keeps on ticking. There are pictures, I think from the tsunami in Japan in 2011, where a building collapsed and you saw a mainframe toppled over in the rubble, but it was still running.

LXF: What about other applications of mainframes?

JM: One of the more interesting workloads that it's really well suited for is blockchain *(see box on page 40)*. Blockchain is all about ledger transactions. Plasticbank

He's probably pretty handy in a fight, come to think of it.

(**www.plasticbank.com**) has really taken the lead here. They're an organisation that's focused on reducing ocean waste. They're out there using Linux on a mainframe for doing all the co-ordination between collecting plastics from third world countries. That's the only way they could do it at scale.

A lot of the traditional applications are still on the rise. Even more banks and financial companies are setting up, especially in South East Asia, China has a big push too. The saying my mainframe colleagues like to quote is that the cloud could go down, the cloud could disappear. (If that happened) it would be tough for us, but we would get by. There would be a lot of inconveniences in life, but we could get by. If the mainframe went away, if all the mainframes in the world shut off, society would plunge into chaos. You wouldn't be able to access money, and all the derivative chaos that comes from that.

Not that we're implying he does fight often, or indeed ever.

LXF: What developments are we going to see coming soon from the Open Mainframe Project?

JM: I think we're working on a couple of different fronts. One is continuing to foster out collaboration efforts within the mainframe ecosystem. Zowe came from a collaboration between IBM, Rocket Software and CA Technologies. They all contributed pieces and said, let's bring this out as open source. There are some other projects we're working on in the wings. ZVM is the hypervisor for mainframes, analogous to KVM. One project currently in incubation is to make it trivial to connect Openstack or Openshift, or something like that, to leverage ZVM as a hypervisor.

Another thing we're just getting more formalised is supporting projects that might want to support a mainframe. The biggest challenge for a lot of those projects is getting hold of a mainframe to test it on, so we're providing mainframe infrastructure for building, CI and testing build environments they can use. We're also helping connect them with expertise if they run into problems and showcasing their stories. It's our goal that, over time, mainframes should be so ubiquitous within computing that it almost doesn't need its own group to promote it. It's not like there's an x86 Foundation, right? LXF

Pop!_OS

Let eyecatching-distro aficionado **Jonni Bidwell** show you the ropes of System76's bespoke Linux flavour.

There have been a few notable efforts to humanise Linux. Today we'd say – and feel free to disagree – that Linux Mint and elementary OS are the most friendly-to-use and all-purpose distros out there. These are of course indebted to Ubuntu for providing a solid foundation and superlative package selection – which in turn owes something to Debian. But it's also arguable that much of this popularity is, or at least was, a result of dissatisfaction with Ubuntu's desktop.

People who didn't like Unity loved Cinnamon, and the people who did like Unity didn't like GNOME 3 (which Ubuntu has used since 17.10). Pantheon, the macOS-like desktop of elementary OS, is attractive not just to fans of fruit-based fashion companies, but to anyone frustrated with over complicated configuration, inconsistently styled applications and ugly fonts. Of course, Windows 10 drives a steady trickle of users to Linux too, as the Start menu begins to resemble some sort of ever growing billboard farm, and updates constantly get in your way.

Hardware compatibility is an important concern too. Today, distros have to cater to all kinds of new-fangled configurations: HiDPI fractional scaling, multi-monitor and hybrid GPU setups, disk encryption. Users want easy access to the latest software, and developers want easy access to their preferred development tools. Providing such features is especially important if you're a manufacturer of Linux systems like Colorado-based System76, and that is why it developed the mysteriously punctuated Pop!_OS.

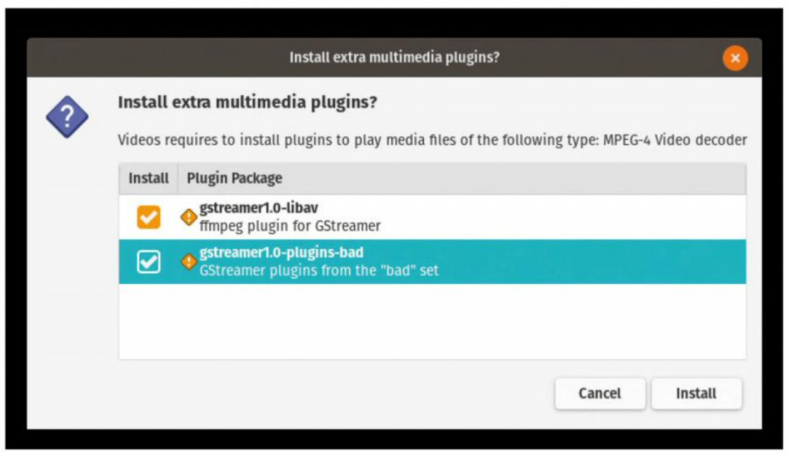
Hopefully by now you'll have had a play with Pop! from the LXFDVD and have seen what all the fuss is about. You may even be itching to install it, in which case check the step-by-step guide on page 49. Before you do though, you should be aware that if you have Nvidia hardware and generally want life to be easier, you should use the Nvidia edition of Pop from https://system76.com/pop. This comes with Nvidia's proprietary driver set up nicely out of the box.

The AMD/Intel edition on the DVD contains only open source drivers, including the Nouveau driver for Nvidia cards, but that's not much use for AAA gaming or CUDA programming. If you have any problems booting the **LXFDVD** or booting a Pop!_OS USB, ensure that you have disabled secure boot in the UEFI settings (often found by pressing F2 or Delete at boot time). Different BIOSes use different hotkeys to invoke a boot menu, but F11 and F12 are popular. Also see our DVD FAQ at **www.linuxformat.com/dvdsupport**. If our DVD works fine for you *(awesome, my first attempt at repacking an initrd – Tech Ed)*, then great: you can skip step 1 in the walkthrough.

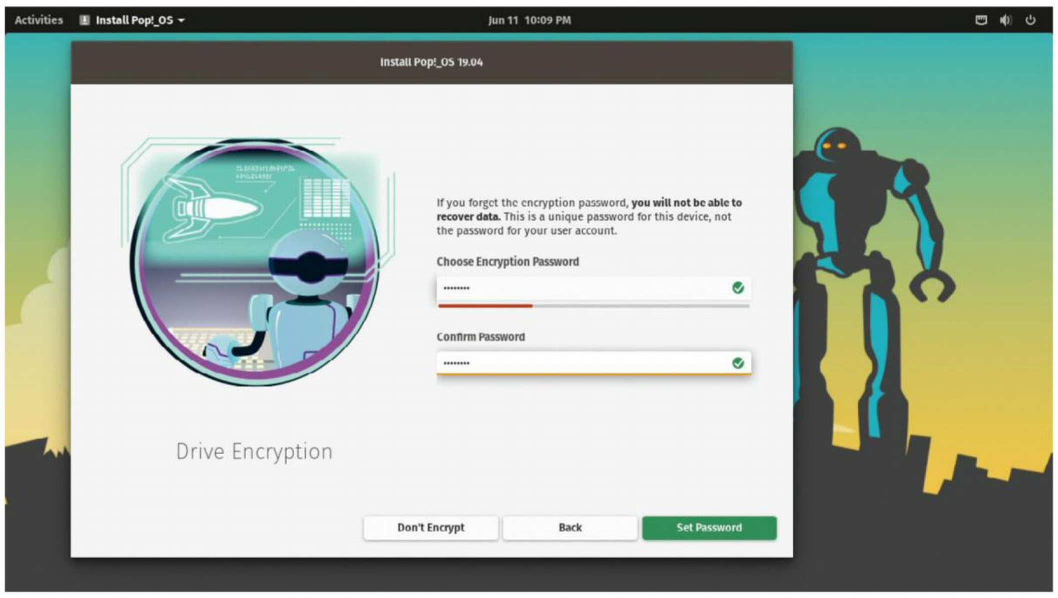
Pop it like it's hot

The guide covers installing alongside Windows; we'd always recommend installing on a different drive if you have one available though. UEFI booting means Windows usually manages to keep itself to itself, or keep its breaking to itself. If you use classic BIOS you might find Windows blithely overwriting the bootloader from time to time, which is annoying, but can be remedied. Either way, it's important to disable Fast Startup in Power Options in the Windows control panel, otherwise you won't be able to boot another OS. If you're setting up Pop as your only OS, you can safely ignore all the partitioning instructions in the walkthrough; just select 'Erase entire disk' from the installer and everything will be set up for you.

You should also be able to safely use the Pop!_OS Installer's partitioning tool to resize the NTFS partition, but it's probably safer to let the devil (Windows) do the devil's work (rearranging NTFS structures). If your EFI partition is less than 512MB – some Windows installs seem to settle for 100MB – then you'll need to use a third-party tool to resize this and possibly move the Windows System reserved partition, which we can't possibly recommend. Hopefully this isn't you and you have no need to worry. *Distint* (Pop's bespoke and delightfully simple installer) will hold your trembling hand throughout the installation process.



The Videos app offers to install any codecs you might need, or alternatively install VLC or MPV.



If you choose to install Pop on its own drive, you can enable full-disk encryption so your data is safe at rest.

Pop!_OS doesn't really resemble Windows or macOS, and it only vaguely resembles Ubuntu's current Gnome setup. But it is easy to get to grips with; hit the Super (Windows) key or click the top left corner to access the activities view, then start typing to find installed applications. That will also show you applications you can install from the *Pop!_Shop* (okay, that does have a nice ring to it) as well as any recently opened matching files. Our initial install weighed in at around 5.5GB, which is pretty modest by today's standards. As with many distributions *LibreOffice* comes bundled, but beyond *Firefox* and core tools there are no crazy-large application inclusions. There are a few gems included, such as the delightful *Geary* email program and System76's own *Popsicle* USB-writing tool. There's

» THE LITTLE THINGS

A lot of what distinguishes Pop!_OS is its behind-the-scenes attention to detail. Things that are traditionally annoying are simple with Pop! System76 has written its own HiDPI daemon, which you can read all about at **http://bit.ly/lxf252hidpi**. It makes light work of whatever complicated display arrangement you're running, and can even manage mixed Hi and LoDPI displays with cunning scaling. Wayland is disabled out of the box, but you can enable it by editing **/etc/gdm3/custom.conf** and commenting the line:

```
WaylandEnable=false
```

You'll then see the usual cog icon when you type in your password at login, and you can choose between Wayland and X.org sessions. Wayland in some multi-DPI situations performs better than X, and in our testing we only ran into a few niggles – mostly using Fedora's experimental *Firefox Wayland* flatpak from **https://firefox-flatpak.mojefedora.cz**.

It's easy to dismiss Pop!_OS as just another Ubuntu clone, but we don't do features on insipid distros, so there must be something to it. Beyond what we've written about here, there are so many other little tweaks to make life easier. These have been done cleverly so you hardly notice them. With users at the heart of its design and that design informed from the unique viewpoint of an OEM, Pop is just lush. It's perfect as a beginner's distro, but it hasn't been 'dumbed down' in any way. It's so flexible and powerful that more advanced users will enjoy it too. Heck, we just installed it on the official **LXF** laptop (*Jonni's Eee PC!–Ed*). So there.

also *Eddy*, which like its rhyming brethren *Gdebi* is a GUI tool for installing DEB files – ideal for the command-line averse.

From the Activities view find the *Pop!_Shop* launcher either by typing it or selecting the Show Applications button on the dock to the right, pausing for a second to admire the bold icon design. You could say that Pop!_OS's initial app offering is quite sparse, but the idea is that you add only what you need to it. Not everyone wants *Visual Studio Code* or *Slack*, but if you do they're very easy to find in the *Pop!_Shop*.

Today's application picks are brought to you by the letter G: *Goxel* – making voxel art is surprisingly calming; *Gnome Twitch* – for watching Gaming On Linux videos; and *GIMP*, still the best way to manipulate images.

CHOOSE YOUR OWN ADVENTURE

"Pop!_OS's initial app offering is quite sparse, but the idea is that you add only what you need to it."

Naturally, command-line installation with *apt* is possible too, and many packages don't have a *Pop!_Shop* entry. One thing missing from the Pop install is the OpenVPN plug-in for NetworkManager, which if you use a VPN (or envisage doing so in the future) you'll want to remedy with:

```
$ sudo apt install network-manager-openvpn-gnome
```

We hope your Pop install doesn't break, but if it does help is at hand in the form of a recovery partition. This essentially lets you boot to something like the Live environment on the DVD, from whence repairs can be carried out. Press Shift and Escape while the system is booting and select the recovery environment from the menu. If that doesn't work you can perform a Refresh Install, which resets all system configuration and applications, but retains your user information and documents, much like the equivalent Windows utility.

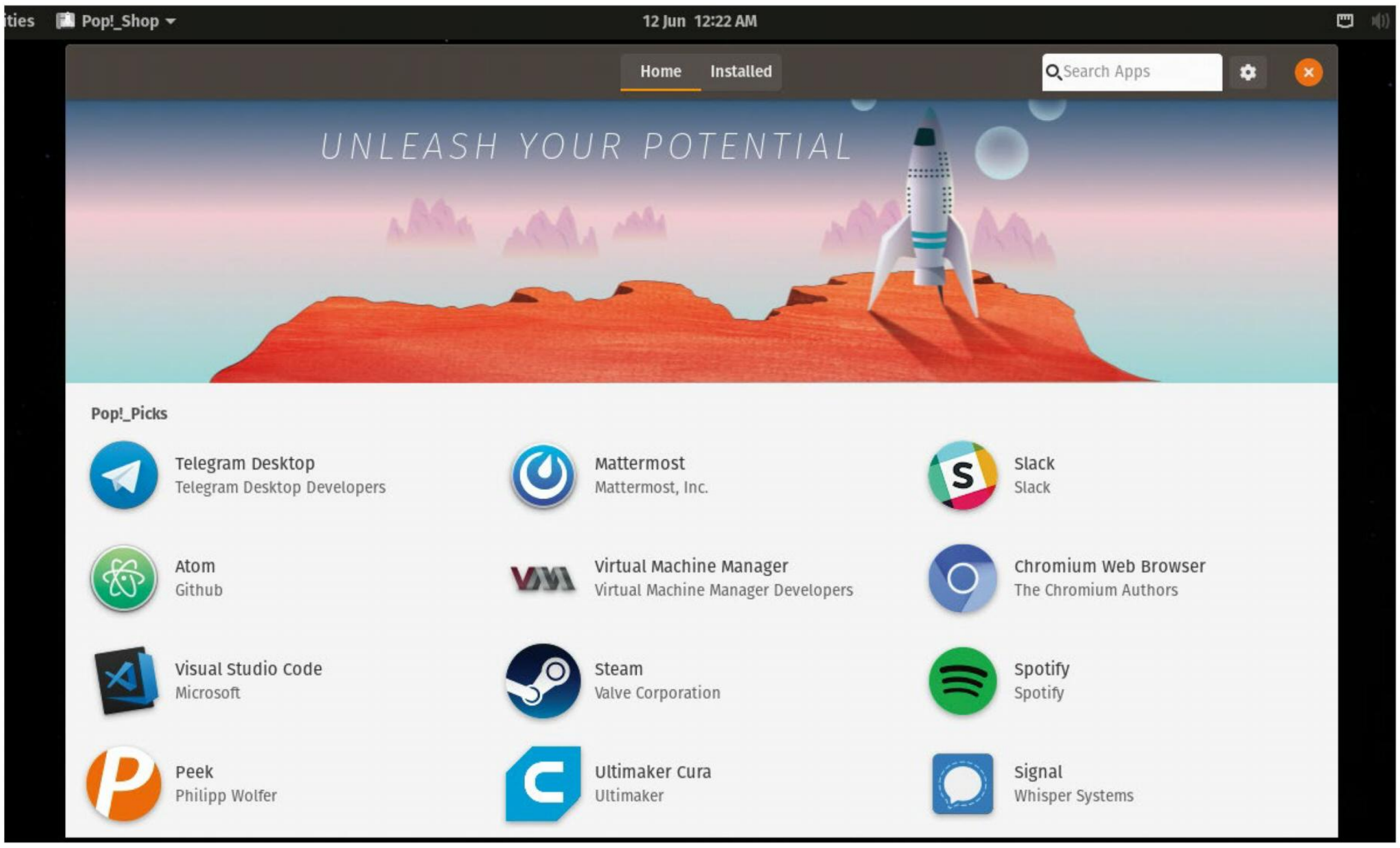
Speaking of booting the system, Pop is pretty much unique among major distros in that it uses *Systemd-boot* – formerly *Gummiboot* – to boot the system on UEFI installs. This means that for once we can say you won't run into any horrible *GRUB* problems on this distro, unless you use classic-BIOS booting. You shouldn't run into any *Systemd-boot* troubles either, by the way, as System76 test all kinds of weird and wonderful configurations on its hardware.

The included video player is fine, but you may prefer to install *MPV*, which handles its own codecs. Once installed it also takes advantage of whatever hardware acceleration is on offer (VDPAU, VA-API or some unholy combination of the twain is configured out of the box) to decode video without bothering your CPU.

The Pop!_OS team has done a great job of preparing documentation too. There's a growing list of guides to common tasks at **https://support.system76.com/articles**. For example, if all the abstruse references to terminal commands throughout this magazine make no sense to you, find out more in the Terminal Basics article there.

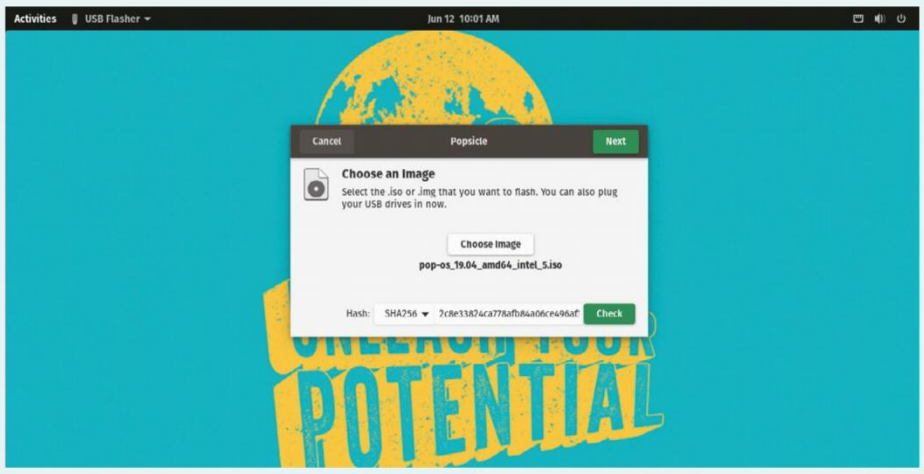
The team are enthusiastic about getting more people involved with development, too. They even mark out specific bugs in their project as being 'bitesize', suitable for aspiring coders to cut their teeth upon. We have to credit them for this, as the growing divide between users and developers will only be lessened if more people get involved, and such outreach efforts are a fine way to encourage that kind of involvement.

We also must credit System76 for its open hardware efforts. Besides contributing to open firmware causes, its flagship Thelio machine includes a custom daughterboard for managing airflow. The specs for this are completely open, and System76 hopes to develop more hardware that follows this model. LXF



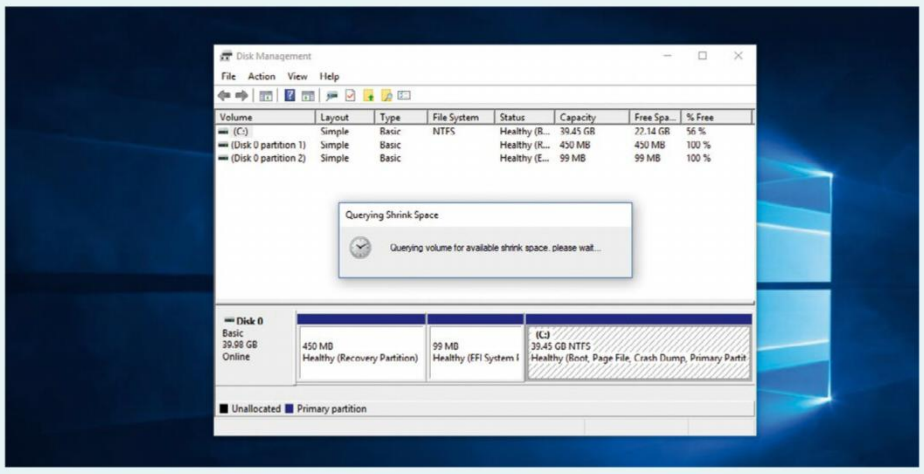
The shopfront offers a good mix of popular proprietary and open source applications.

INSTALLING POP! ALONGSIDE WINDOWS



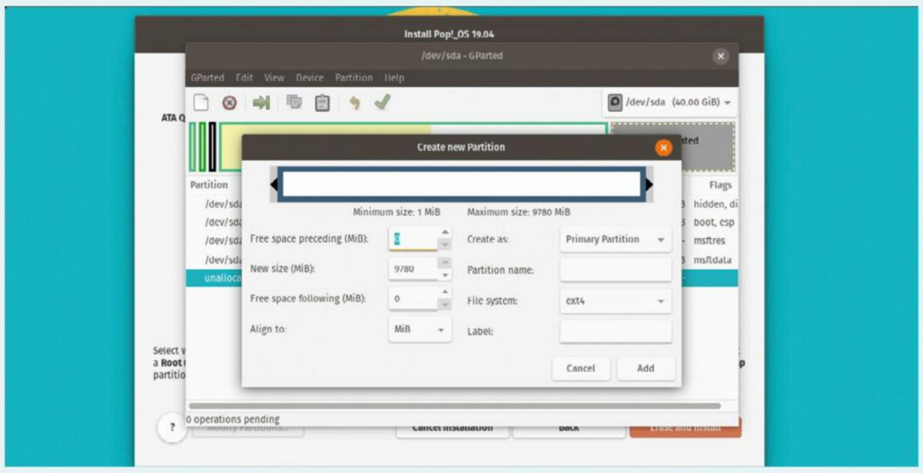
1 Write an install medium

If you have an Nvidia graphics card you'll want to fetch the Nvidia build from **https://system76.com/pop** and burn it to a DVD or write it to a USB stick. If you have trouble booting Pop!_OS from the **LXFDVD** (or have no DVD drive), you should write the ISO file from the **Pop/** directory (or the link above) to a USB.



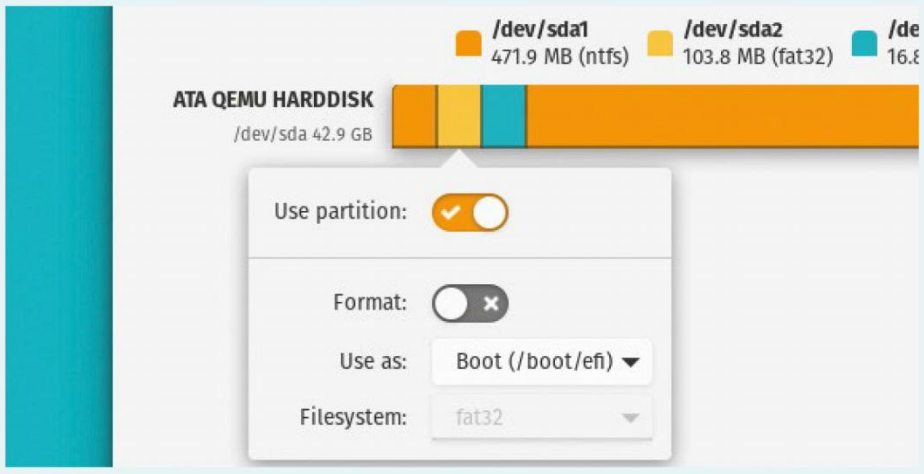
2 Make space

Start windows and open the Disk Management tool. The easiest way to find it is by opening the start menu and typing the first few letters of 'disk man'. Right-click the C: drive and choose Shrink Volume. Make some space for Pop. Pop recommends at least 20GB, but if you can spare it, use it.



3 Boot Pop!

Boot Pop!_OS and start the installer. After answering some localisation questions choose the Custom (Advanced) option, unless you're happy to get rid of everything on the target drive. Select 'Modify partitions', right-click the Unallocated Space and click New. Go with the defaults and then click Add and finally Apply the changes.

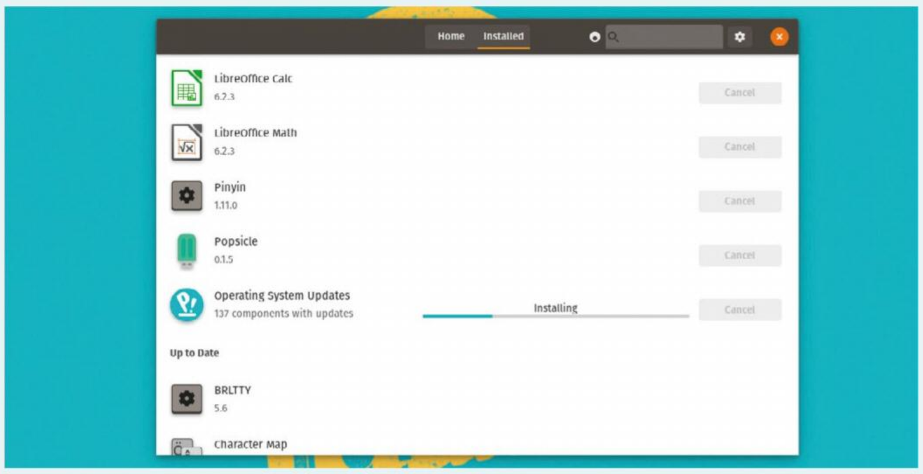


4 Select partitions

Exit the partitioning tool and click the new partition. Click the 'Use partition' switch, again leaving the defaults. Click the EFI partition (it's usually second from the left on a Windows install and is FAT32-formatted) and again use it as a Boot partition, but please don't format it as that will break Windows.

5 Lift off

Hit Erase and Install to begin the fun. Either watch the progress bar intently or make a cup of tea. When the installer's done hit Reboot and do some final setup steps. You can enable location services and connect online services. Finally, set up a user and password, then click the big tick to start using Pop!_OS.



6 Update and augment

Updates will probably be available. You should install these when prompted, then switch from the Installed tab to the Home tab. Here you can peruse the multitude of applications awaiting your attention in the Pop!_Shop. Pop is deliberately lean out of the box, so you may want to install some workhorse applications at this point.

Les Pounder works with groups such as the Raspberry Pi Foundation to help boost people's maker skills.

» MICROPYTHON

Over the last few days, PyCon 2019 – the largest Python conference in the world – has just taken place in the USA. Sadly I was unable to attend this year, but I have been keeping an eye on the events via social media.

This year Python on hardware, more commonly known as MicroPython, seems to be the buzz. MicroPython was created six years ago by Damien George and the first commercial device to support it was his PyBoard. But in 2019 there are a myriad of devices that can run MicroPython – such as micro:bit, ESP8266 and ESP32 and many others. In 2017 Adafruit released its fork called CircuitPython, designed initially for use with its boards, but which has since been ported to many others.

Is it the end for the Arduino? No: the Arduino is still king of the microcontroller scene and that will be the case for a few years yet at least. So why is Python important for the microcontroller? Well, it comes down to flexibility, in that the syntax of MicroPython is very similar to Python 3; in fact, it *is* a version of Python 3 for just this use. This means that existing Python programmers can 'just get building' their hardware projects. Python is a remarkably flexible language and that is evident by the sheer volume of projects that have appeared since the Raspberry Pi first arrived in 2010.

The Pi is the pioneer for Python with electronics. Without the popularity of Raspberry Pi and the quick adoption of Python as its de-facto language, we would not have seen MicroPython.

Raspberry Pi protecting worldwide wildlife

Meet the group developing Pi-powered monitoring to help protect endangered species.

From tracking penguin colonies in Antarctica to elephants in India, the challenges involved with the expansion of human communities and the destruction of wild habitats means conflict between people and wildlife are ever increasing. It turns out that developing smart monitoring tools (often Pi-based) can help! One such team pioneering this is the Arribada Initiative, which has developed a wide range of monitoring devices deployed in all sorts of environments. One high-profile project is Penguin Watch hosted at Zooniverse (**http://bit.ly/lxf252penguin**), which deploys Pi-powered timelapse cameras to monitor and tag colonies through a citizen science initiative. Having developed a ruggedised, solar-powered, satellite-connected system, these cameras were deployed and happily survived the Antarctic winter.

The Raspberry Pi range provides the perfect base to develop devices that cover a wide range of monitoring activities and the wide range of environmental challenges that are faced by monitoring systems. For more see **http://blog.arribada.org** and get code for projects at **https://github.com/IRNAS**.

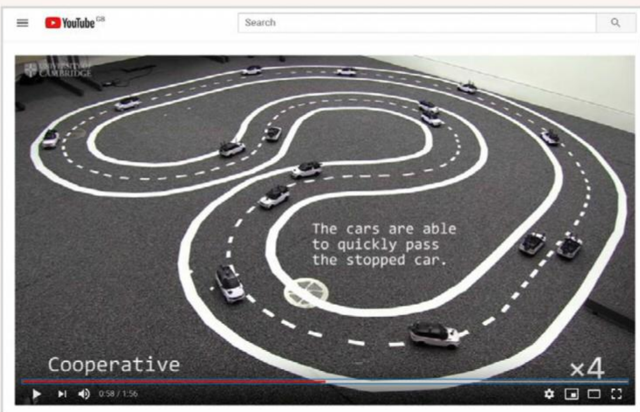
CREDIT: https://github.com/IRNAS/PitStop1 (CC BY-SA 4.0)

Raspberry Pis are even finding their way onto the backs of turtles.

Traffic modelling

Real-world Pi-based testing.

Testing autonomous vehicles in the real world can be expensive and dangerous. So the smartypants at Cambridge University are using Pi-driven model cars to do the same thing on a tiny scale! Turns out autonomous cooperative driving makes traffic flow far more smoothly – and no one got hurt testing it all out. Want to know more: **www.proroklab.org**.



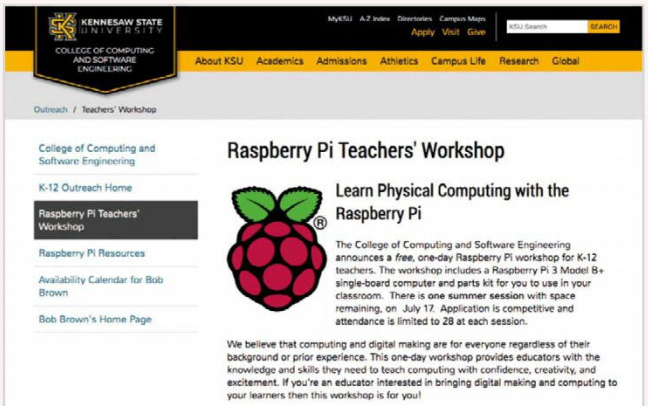
This image gives Effy commuter-related panic attacks.

CREDIT: www.youtube.com/watch?v=e0LIU1Sf6p0

EduPi USA

A model for success.

The Pi Foundation's Blog has a fascinating report on how a retired teacher in the US has had storming success introducing Pi-based training programmes. Dr Bob Brown, a Pi Certified Educator and Picademy attendee, used his experience to kick-start a series of highly successful workshops. Inspirational stuff. Find out more: **http://bit.ly/lxf252edupi**.



A perfect way to introduce coding, tech and the Pi.

CREDIT: https://ccse.kennesaw.edu

4tronix PiBug 2WD

Les Pounder takes a look at a simple robot that can be built in less than 30 minutes without any special tools.

IN BRIEF

Designed for the Raspberry Pi A, A+, B and B+ boards, PiBug is a small and easy to build robot that uses the Raspberry Pi board as a central part of the chassis. No soldering or expensive tools are required – just 30 minutes and a screwdriver.

Robots and the Raspberry Pi go hand-in-hand. There are countless kits on the market and all are vying to be the 'best' or 'easiest' to build and use. 4tronix has a long history of building robots specifically for the education market, and with PiBug we see their latest robot in its growing army.

PiBug is a little different to other robot builds. Rather than include a chassis onto which we connect our Raspberry Pi, the Pi becomes a major part of the chassis. Coming as a collection of three circuit boards, motors and screws, PiBug is held together via a series of brass columns that connect through the four screw holes around the edge of the Raspberry Pi.

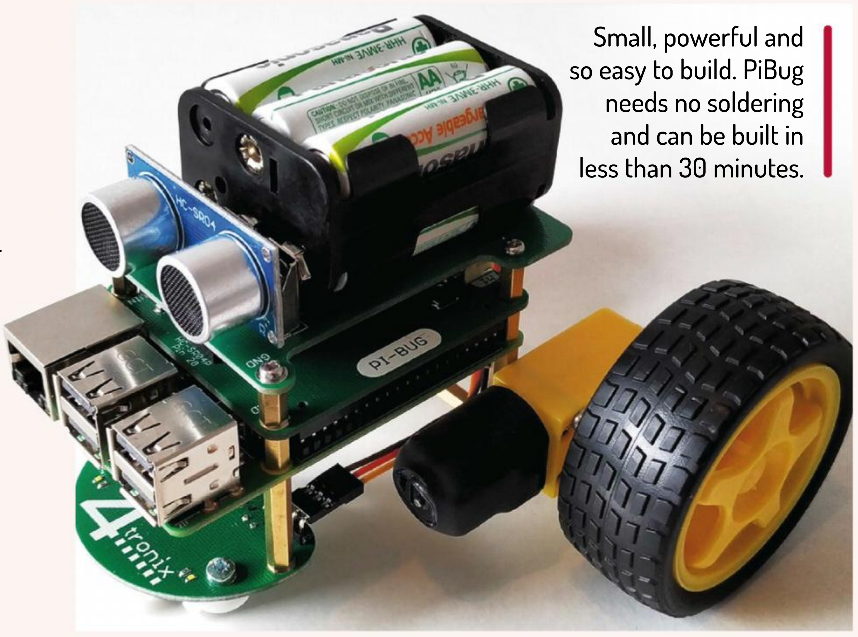
Via a short series of instructions on the website we were able to build our robot in less than 20 minutes, including the wiring of two extra sensors (line following and ultrasonic.) The build required nothing more than a screwdriver and a ruler to measure the length of components. No soldering is required and the kits can be broken down into their parts for reuse in class.

PiBug is designed for Raspberry Pi models A, A+, B and B+. That means your Zero or original Pi will not fit, unless you are determined and have some spare parts. Powering the Pi and the robot are six 1.2V rechargeable batteries, giving us 7.2V of power. This is the recommended configuration, as 1.5V alkaline batteries may introduce a reset bug where the Pi resets when making a turn or rapid change in direction.

The batteries are stored in a holder atop the robot, with a connector to the board locking them into place. Just make sure to use the included Velcro strap to hold the batteries in, as a crash dislodged our batteries, launching them across the room.

The two DC motors are controlled via a DRV8833 motor controller, which is part of the middle PCB in the sandwich. This controller is a staple in the robot community as it offers ease of use at a low cost – a great entry-level motor controller.

Software for the PiBug is provided via an install script on the 4tronix website, and while this may not be the most secure method of installing it, the code is fully examinable and installation is transparent to the user. Once installed we have a directory called **pibug** containing the library **pibug.py**, which has all the functions needed to control the motors and react to sensor input.



Small, powerful and so easy to build. PiBug needs no soldering and can be built in less than 30 minutes.

The library is written in Python 2, which is a shame as this will lose support in 2020. The code will work with Python 3, but you need to make a few changes to `print` statements, as in Python 3 they are written as functions with parentheses. Once this bug is ironed out the **pibug.py** library works well, and we can clearly see how to retrieve and store data from our sensors and control the direction and speed of the motors.

The PiBug costs £24 for the basic kit. Adding the extra sensors (which we used) takes the price to £32.15, which is a good buy considering how sturdy and easy to build it is. This is a ready-to-go solution for classrooms and those wanting to get started with robotics.

More advanced roboteers will need to look elsewhere, but as the target market for this robot is educational, 4tronix has hit the nail on the head! This is an ideal first robot that offers cost-effective, reusable parts. LXF

Close up of the clever "no chassis" construction method.

VERDICT

DEVELOPER: 4tronix
WEB: https://shop.4tronix.co.uk
PRICE: From £24 (Model reviewed £32.15)

FEATURES	8/10	EASE OF USE	10/10
PERFORMANCE	9/10	VALUE	8/10

Ideal for the classroom, low-cost and easy to use, a solid little robot that offers plenty of functionality in a unique package.

» Rating 8/10

EXPLORER HAT PRO

Using analogue HAT interfaces

Les Pounder shows us two more projects using the deceptively simple Explorer HAT Pro – and a little Python code.

OUR EXPERT

Les Pounder is a freelance maker who works with organisations such as the Raspberry Pi Foundation to promote maker skills.

This issue we'll learn how to use the Explorer HAT Pro to work with analogue electronics by creating two projects. First we create a simple light-detecting nightlight, then we create an electronic candle that we can blow out. But first we need to set up our equipment.

With the power turned off to your Raspberry Pi, place the Explorer HAT Pro so that it connects to all 40 GPIO pins, and so that it fits neatly above the Pi. Now connect your keyboard, mouse, HDMI and so on to your Pi and boot up. For the next step you will need to connect to the internet using Wi-Fi or Ethernet. Once connected, open a terminal, the icon for which is found in the top left of the screen. In the terminal type the following to automatically install and configure the Explorer HAT Pro software:

```
$ curl https://get.pimoroni.com/explorerhat | bash
```

During the install you will be asked to configure I2C and to install example code. Answer yes to these questions. Once installed, close the terminal window.

Project 1: Nightlight

To introduce analogue electronics we shall use the most simple device, a light dependent resistor (LDR). This component changes resistance based on light levels. For this project we shall place the LDR on the breadboard and connect one leg to 3.3V, and the other to GND via the 22K ohm resistor. Then we connect Analog 1 from the board to the same line as our resistor and LDR. We do this as it creates a voltage divider that we use to measure the light level as a voltage. See the diagram in the download for an accurate reference.

We now move on to the code. Using your favourite editor (*IDLE*, *Mu*, *Thonny*) create a new file and then save that new file as **nightlight.py**. Remember to save often! We start the code with two imported libraries, `time` and `explorerhat`, which we rename to `eh` for better convenience.

```
import time
import explorerhat as eh
```

Then we create a loop to constantly run our code.

```
while True:
```

Let's take a light level reading and store it inside a variable called `light`. This will show the voltage that our circuit is producing based on the resistance level of the LDR. This is printed to the Python shell for debug.

```
light = eh.analog.one.read()
print(light)
```

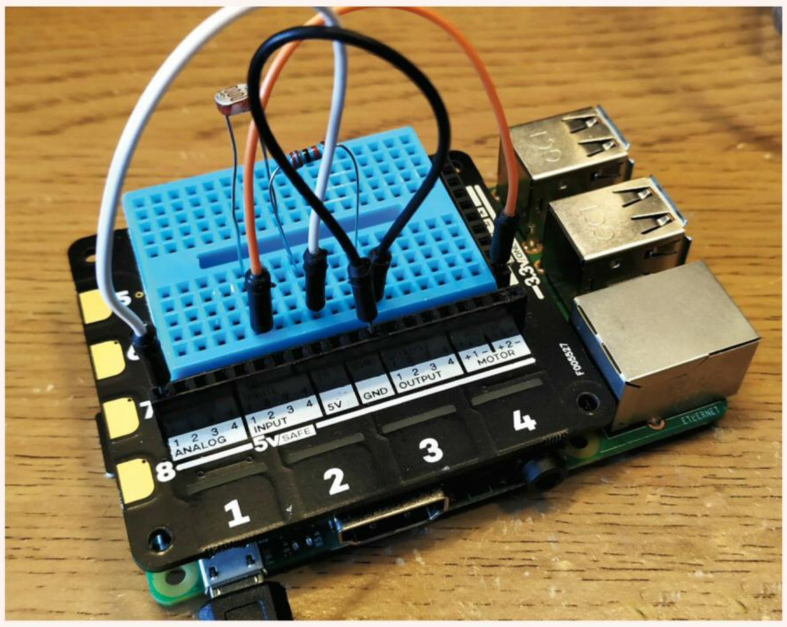
We then use a conditional test that checks to see if the voltage detected is less than 2V. If that is the case then it must be getting dark, so the four LEDs on the Explorer HAT Pro will turn on.

```
if light < 2.0:
  eh.light.on()
```

But if it daylight then the voltage will be over 2V and so the `else` condition activates and turns off the Explorer HAT LEDs.

```
else:
  eh.light.off()
```

Save the code and when ready, run it. Cover the LDR with your hand and the LEDs should light up; move it and the LEDs turn off. If you need to tweak the values, take a look at the Python shell to see what voltage changes there are when the sensor detects darkness.



This simple circuit offers a great introduction to analogue electronics. It requires only a few cheap components and is accessible to all.

Project 2: A digital candle

This project is inspired by the Nintendo DS, specifically a section in *The Legend of Zelda: Phantom Hourglass* where you are instructed to 'blow out' fire/torches in the game. To do this we blow on to the console and the game reacts. In reality we are blowing onto a

YOU NEED

- Any 40-GPIO-pin Pi
- Latest Raspbian
- Explorer HAT Pro
- Electret microphone board
- 3x male-to-male jumper wires
- Light dependent resistor (LDR)
- 22k ohm resistor (red, red, orange, gold)
- Code: http://bit.ly/lxf252pihat

microphone and an analogue signal is being measured to act as a trigger. To replicate this we shall control the LEDs present on the Explorer HAT Pro using a microphone board. The microphone is connected to our Explorer HAT Pro as: GND > GND, OUT > Analog 1 and VCC > 5V.

We now move on to the code. Create a new file as before and then save that new file as **candle.py**. The first three lines of our Python code import prewritten libraries of code. The first is called `time` and we use that to control the speed of our project. Next we import the `explorerhat` library, but rename it to `eh` for ease of use. Lastly we import the `random` library which will be used for our candle flicker effect.

```
import time
import explorerhat as eh
import random
```

To run our code we need a loop and for that we shall use `while True`, which will run our code until we stop it.

```
while True:
```

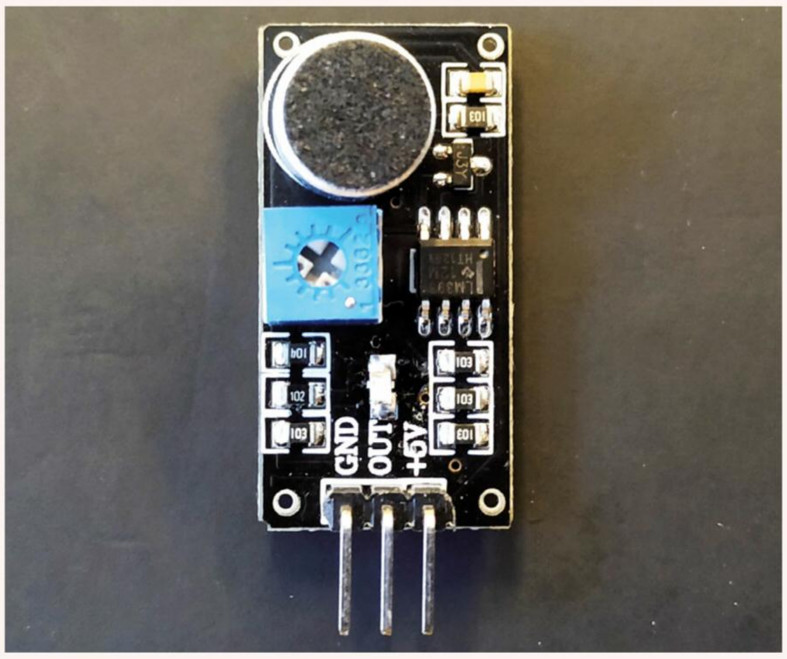
The following code is indented to show that it runs inside of the loop. We create four variables, called `on`, `off`, `fade_in` and `fade_out`. These variables will be later used to control the flicker effect of the LEDs. But how do we create that? Well, we need to introduce a little randomisation. Using the `random.uniform` function we can generate a random float (a number with a decimal point) between 0.0 and 1.0, and then store it in the variable:

```
    on = random.uniform(0.0,1.0)
    off = random.uniform(0.0,1.0)
    fade_in = random.uniform(0.0,1.0)
    fade_out = random.uniform(0.0,1.0)
```

To detect that someone has 'blown out' the candle, we need to read the sensor connected to Analog 1. Using the Explorer HAT library we read the value of the sensor and store it as a variable called `blow`, which is then printed to the Python shell for debug with a small delay for legibility.

```
    blow = eh.analog.one.read()
    print(blow)
    time.sleep(0.1)
```

If we run the code as is, it will print the sensor



The microphone/sound sensor has a blue trim pot that can be used to alter the output voltage. This is how we can calibrate the sensor for the environment.

reading, which in our case was 5.00. This is the voltage from the sensor. Using the trim pot (blue dial) on the microphone, we can tune the sensor; we chose 1.5 as the trigger point. So if the voltage goes over the trigger point it means that a loud sound (blowing out the candle) has been detected.

```
    if blow > 1.5:
```

Now we need to turn the Explorer HAT lights off. Using the Explorer HAT library we turn off all the lights and then wait for two seconds before the loop repeats.

```
        eh.light.off()
        time.sleep(2)
```

But if we haven't blown out the candle, the `else` condition activates and it uses the `pulse` function to make the lights flicker using the `on`, `off`, `fade_in` and `fade_out` values.

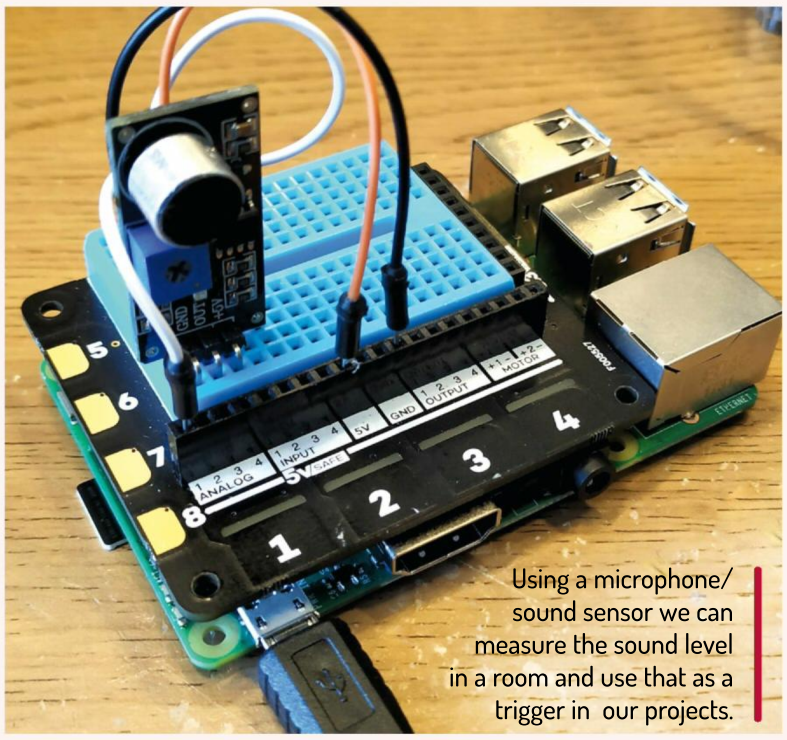
```
    else:
        eh.light.pulse(on, off, fade_in, fade_out)
```

Save the code and then run it. Now try to blow out the candle. If it works, great; if not, adjust the values of your code and the trim pot on the microphone.

So there we have it, two simple projects from one great board. LXF

QUICK TIP

Need to quickly check the sensor data? Open the Python shell and type 'import explorerhat as eh' then 'eh.analog.one.read()'. This is really useful when calibrating the sensor before general use.



Using a microphone/sound sensor we can measure the sound level in a room and use that as a trigger in our projects.

» EXPLORER HAT PRO

We have long been fans of this board because it packs so much into a tiny package. In this tutorial we used its analogue inputs to read voltage levels. But we also have four protected inputs, rated up to 5V, and four outputs rated for 5V. There are two motor connections for use with DC motors, so we can easily build robots.

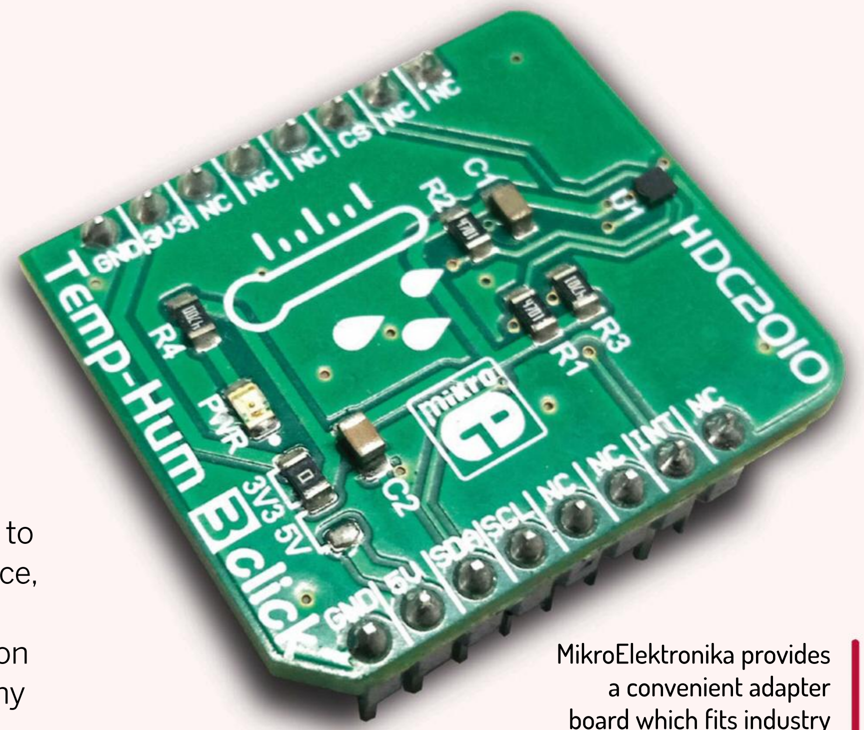
We have four touch buttons that can be used to trigger projects to life. We also have four capacitive touch connections for use with crocodile clips, enabling us to make connections and circuits that react to human skin. There also four LEDs that we used in these projects – these can be on or off, or we can fade their light levels to create subtle glows.

On the right side of the board are connections for I2C and SPI (3.3V only) which means we can connect other sensors and components using their protocols. Lastly, we have a breadboard built into the board and ready for use to build our projects upon it. We get all of this versatility for around £20 – a bargain when you consider how much we can learn from it.

INTERFACES

Monitor humidity with the I2C bus

Tam Hanna feels moist and no one likes that, so he's monitoring his environment.



MikroElektronika provides a convenient adapter board which fits industry standard breadboards.

OUR EXPERT
Tam Hanna whiles away the weekends monitoring his new basement HQ, crafting items on his CnC milling machine and plotting murder most fowl for his tea.

Living under the ground has benefits and disadvantages. While the person who treats himself to a subterranean living space starts to appreciate an absolutely noise-free sleeping experience, managing underground real estate challenges even experienced landladies such as my wife. The inspiration for this story struck when she had to travel to Germany for business: she wanted to keep an eye on the humidity and temperature levels of our headquarters.

The development of the semiconductor industry has led to smart sensors which combine a sensing element with conditioning logic. Output is handled via hardware buses such as I2C, the bus which we'll use in the following tutorial. The Texas Instruments HDC2010 is an excellent temperature sensor; it not only takes care of temperature, but also keeps an eye on humidity. All that is done with a pretty impressive accuracy of within two per cent. While this might not sound like much, it's the absolute high end of what is currently possible in affordable semiconductor sensors.

Sadly, Texas Instruments makes the HDC2010 in an extremely small package. Soldering this by hand is impossible, and reflowing it with the normal reflow oven is difficult at best. The official evaluation kit from Texas Instruments also is quite pricey, leaving unexperienced designers in a bit of a rut.

Temp&Hum ahoy

Fortunately, MikroElektronika recently started devoting resources to solving this problem. In the case of the HDC2010, the solution goes by the name of Temp&Hum 3 Click and can be yours for about $13 from **www.mikroe.com/temp-hum-3-click**.

Freshly downloaded versions of Raspbian usually disable the Pi's I2C interface. Fortunately, solving this problem is not hard – open *raspi-config*, switch to Interfacing options and enable the I2C interface. After the obligatory reboot, the I2C interfaces show up in the device tree:

```
pi@raspberrypi:/dev $ ls | grep "i2"
i2c-1
```

Scanning for the presence of devices is best accomplished via a small utility called *i2cdetect*. It's not part of the standard distribution, so download it from the package repository as follows:

```
pi@raspberrypi:/dev $ sudo apt-get install i2c-tools
```

After that, it's time to assemble the actual circuit: see the diagram on page 56. Experienced electrical engineers might wonder why there is no pull-up resistor present. The answer is found in the schematics of the MikroElektronika board *(see page 57)*: the company put a set of pull-up resistors in place.

Let's take a quick look at how I2C actually works. The bus, originally developed by Philips for various hi-fi applications, consists of the master and a set of slaves. The SCL line – short for serial clock – is toggled by the master. It is responsible for setting the speed at which the entity of the bus operates. SDA – short for serial data – is modulated by master or by slave in order to enable actual data transmission.

Master and slave can communicate over one line due to the use of the open drain principle. The above-mentioned pull-up resistors lazily pull SCL and SDA to the positive voltage level. Both master and slaves can pull the line down via a transistor – the interesting bits of this technique are explained in the box on page 55.

For us, however, the next step involves finding out if the Raspberry Pi is able to detect the HDC2010. This is accomplished by *i2cdetect* - the parameter `-y` eliminates the highly annoying warning about potential problems which might occur during the scan process:

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0 1 2 3 4 5 6 7 8 9 a b c d e f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Should your *i2cdetect* run not show a slave presence at address 40, check the wiring. Furthermore, use an oscilloscope to verify the presence of waveforms on SCL; should a logic analyser be on hand, you can also use it to find out more.

Let's code!

At this point, we need to start coding. In theory, opening the data sheet and taking a careful look at it is the action of choice. In practice, coding a driver from scratch tends to be unproductive; almost all semiconductor vendors provide example drivers. In the case of Texas Instruments, simply visit **http://bit.ly/LXF252i2c** and look for the string 'snac075' to download a sample driver for the Arduino platform.

When dealing with manufacturer drivers, that process usually consists of adapting their logic to the API at hand and removing needed parts. For some reason, manufacturers' application engineers insist on the implementation of every useless feature. Like most Arduino examples, the example code provided by Texas Instruments consists of three files. In addition to the header and a **.cpp** file, the semiconductor vendor provides a **.ino** file containing an example demonstrating the API. For convenience's sake, open all three of these files in an editor.

Texas Instruments did an extraordinary job in encapsulating the Arduino-specific parts of the API from the rest of the logic. Because of that, we'll reuse the class provided; in practice, cutting the logic out can often end up being quicker.

Start out by recreating the header file on Raspberry Pi's memory. The contents can be taken one by one from the original file – just make sure to remove the two inclusions of Arduino-specific header files, as they will not be available when compiling C++ code for the Pi. Porting the body of the class is accomplished by using the process set out in the book *Code Reading: The Open Source Perspective* by Diomidis Spinellis. He recommends that code should be considered compilable: make the compiler read it, and use the list of errors to find a working solution.

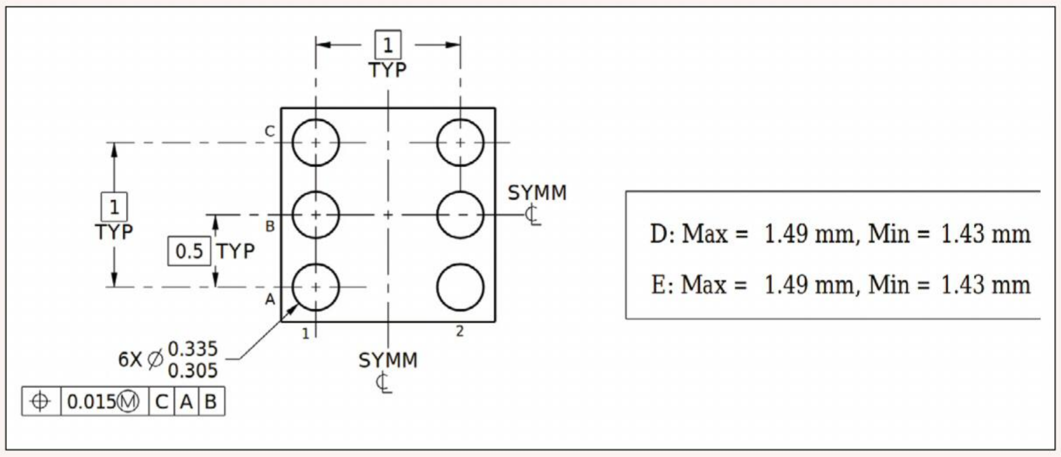
The first step for porting the **.cpp** file involves adjusting its inclusions. Obviously, the Arduino-related library include must be replaced with one intended for the Raspberry Pi:

```
#include "HDC2010.h"
#include <wiringPiI2C.h>
```

On the Arduino, all header files become part of the system library; in the case of our local compilation process, the file must instead be loaded from the current working path. We are ready to order the first compile run:

```
pi@raspberrypi:~ $ gcc HDC2010.cpp -lwiringPi
...
HDC2010.h:98:3: error: 'uint8_t' does not name a type
```

When done, the compiler emits a list of more than a few dozen errors. Glance over them to find low-hanging fruit to eliminate first; eliminating simple problems reduces the length of the error list. We decided to go after those related to the various `uint` types first. Eliminating these first clears the 'fog of war' surrounding the codebase at hand.



The HDC2010. is small and placing it by hand is almost impossible, as the lands on the PCB are really tiny.

A classic nuisance when moving code between platforms involves variable declarations. On the Arduino, a set of types for specific integers is declared to simplify interfacing hardware to code. The Raspberry Pi, obviously, does not have these types, as the system usually is not used for direct hardware interfacing.

Fixing the problem is easy. Open the header file, and add the following defines to bind the tags to variable declarations:

```
#ifndef HDC2010_H
#define HDC2010_h
#define uint8_t unsigned char
#define uint16_t unsigned short int
```

Running the new version of the code leads to a significantly reduced number of errors; another run of *GCC* shows the problems that still need to be tackled.

A careful look at the warning messages shows us that Texas Instruments implemented a set of functions to encapsulate the Wiring API used for I2C communications from the rest of the driver. We need to look at the functions responsible for hardware interaction, and the rest of the code can be used as-is.

Let's start our work with the `begin` function that is responsible for establishing a connection. In case of the Wiring API, creating a connection to the I2C engine is

QUICK TIP

Underground locations can have humidity problems: air, like all gases, can hold a specific amount of water depending on its temperature. As the temperature falls, it holds less humidity – pumping dry, hot air into a dry but cold space always leads to condensation.

» A QUESTION OF (LIMITED) RANGE

One interesting problem of the 'open drain' architecture involves the behaviour of the bus under stress. While the low transition normally is sharp, the charging of the line takes place according to the exponential function known from capacitors.

The capacity of the cable increases with its length and puts limits on the length of I2C buses. When Philips designed the standard, its engineers never expected the bus to extend beyond the confines of a small printed circuit board.

Should you ever find yourself in the situation where a high-range I2C bus is needed, multiple venues are available. First of all, the value of the pull-up resistors can be reduced so that more power is available to charge the 'cable capacitor'. This approach, however, has its limits: keep in mind that the master and the slaves need to sink the current supplied.

Another approach involves reducing the speed of the bus. Most sensors can work with almost any speed ranging from 100 to 400kHz. The slower the bus, the more time is thus given to the resistors to restore the line to the positive voltage level between data exchanges.

simple. Sadly, the wiringPi API takes a different approach in that it returns a file descriptor for the interface. Due to that, we need to add a member to the class definition which holds the file descriptor:

```
      private:
            int myFD;
            int _addr;  // Address of sensor
```

Once this is out of the way, the `begin` method can be adjusted so that connections can take place:

```
void HDC2010::begin(void)
{
      myFD = wiringPiI2CSetup (0x40) ;
}
```

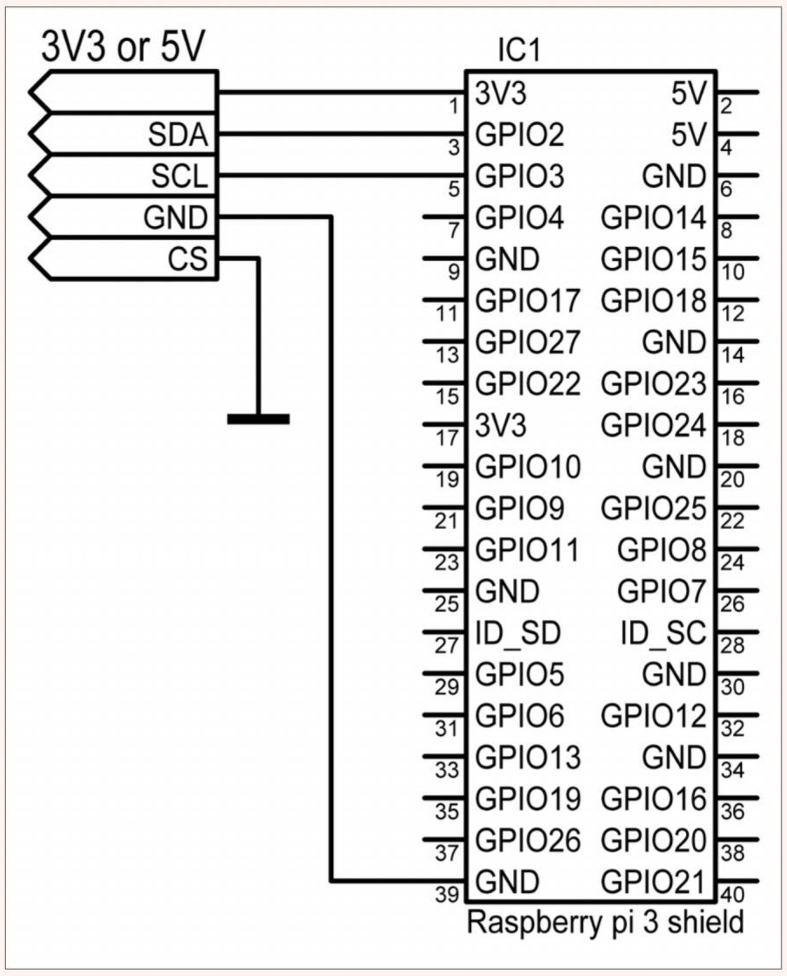
Purists might complain about our practice of not checking the return value carefully; while checking for `-1` is recommended in production code, we omitted it here for clarity reasons.

While finding the best approach to navigate the compile errors is both art and science, we consider the `reset()` function to be the next valuable target:

```
void HDC2010::reset(void)
{
      uint8_t configContents;
      configContents = readReg(CONFIG);
      configContents = (configContents | 0x80);
      writeReg(CONFIG, configContents);
      delay(50);
}
```

While `reset()` looks free of accesses to the Wire API, the use of the `delay` function – normally part of the wiringPi library – causes issues. We limited ourselves to including the header dedicated to the I2C library; add the following declaration to eliminate another problem:

```
#include <wiringPi.h>
```



Connecting the HDC2010 to the Raspberry Pi is not difficult.

A question of registers

While the I2C bus can, in theory, transmit just about any kind of information, most parts work on a register basis. The internals of the part to behave like a classic key value store, which can be addressed, written to and read from by the master.

On the Arduino's I2C API, writing values is a two-step process. The register first needs to be opened for modification, after which the task at hand can be done. In the case of our driver, the design pattern is manifested in function `openReg`:

```
void HDC2010::openReg(uint8_t reg)
{
  Wire.beginTransmission(_addr);  //Connect HDC2010
  Wire.write(reg);  // point to specified register
  Wire.endTransmission();   // Relinquish bus control
}
```

While `openReg` is not needed on the Raspberry Pi, removing the function completely would require significant rewriting of our code. A more comfortable way involves removing the entity of the code and replacing it with no operation comment:

```
void HDC2010::openReg(uint8_t reg)
{
  //NOP - No OPeration on RaspBian
}
```

Readin' and writin'

Now that the individual activation of the bus is out of the way, we need to proceed to reading and writing register information. The original writing function for the Arduino is complex, as the Wiring API requires you to send the register and the data bytes separately:

```
void HDC2010::writeReg(uint8_t reg, uint8_t data)
{
  Wire.beginTransmission(_addr);  // Open Device
  Wire.write(reg);  // Point to register
  Wire.write(data);  // Write data to register
  Wire.endTransmission();  // Relinquish bus control
}
```

I2C bus transactions can involve either 8- or 16-bit values. A careful look at the types used in the methods of Texas Instruments' example show us that the HDC2010 is purely an 8-bit device. Consulting the documentation of the wiringPi API reveals the presence of a dedicated function intended for writing 8-bit

» HUNT DOWN SERIAL PROBLEMS

When selling high-end oscilloscopes, sales people usually jump back and forth about the variety of protocol decoders implemented a model. Their obvious benefit is that the logic content of the waveforms can be shown alongside the physical voltage information gained: you don't only see the actual voltage levels, but also get some kind of interpretation in an overlay or a window shown at the side of the waveform display.

Being admittedly old-fashioned, we don't like this – for a simple reason – experience shows that issues usually are not caused by electronics, but by software or other logical problems such as a misunderstood datasheet or a mistake in the reference driver. Tracking these down usually requires you to look at your code – which lives on the workstation and not on the oscilloscope.

USB logic analysers such as the Saleae tend to be more convenient in this operating mode. Not only are they cheaper than most high-end scopes, the output can be shown comfortably on your workstation's screen right next to the code in question. Finally, also keep an eye on PicoScope: its extremely powerful decoder API lets you adapt the product to new protocols on demand.

register values. This allows us to create a replacement:

```
void HDC2010::writeReg(uint8_t reg, uint8_t data)
{
  wiringPiI2CWriteReg8(myFD, reg, data);
}
```

Reading, in principle, is done along the same lines. The Arduino API shows itself to be extremely verbose:

```
uint8_t HDC2010::readReg(uint8_t reg)
{
    openReg(reg);
    uint8_t reading;  // holds byte of read data
    Wire.requestFrom(_addr, 1);  // Request 1 byte
    Wire.endTransmission();  // Relinquish bus control
    if (1 <= Wire.available())
    {
        reading = (Wire.read());  // Read byte
    }
    return reading;
}
```

As we dealing with 8-bit values, the `readReg` function can also be greatly abbreviated:

```
uint8_t HDC2010::readReg(uint8_t reg)
{
    return wiringPiI2CReadReg8(myFD, reg);
}
```

Save the changes to the file and order another compilation process – it will still fail, but with an extremely interesting message:

```
pi@raspberrypi:~ $ gcc HDC2010.cpp -lwiringPi
(.text+0x34): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

When compiling a C++ program not intended to be a library, the C++ standard requires the presence of the `main` function – so far, we've focused on simply porting the classes. It's now time to create another file and start by including the various headers required:

```
#include <stdio.h>
#include "HDC2010.h"
#include <wiringPi.h>
```

After that, the actual `main` function can be created:

```
int main()
{
  HDC2010 sensor(0x00);
  sensor.begin();
  sensor.reset();
```

Adapting the example to an C++ environment is made difficult. Most importantly, C++ code living in the Arduino environment gets invoked in relatively complex ways. Due to this, our instance of the driver class is obliged to live on the stack.

After creating it, we invoke both the `begin` and `reset` methods. Invoking `reset` might look superfluous at first glance, but makes good sense – when interacting with sensors, ensuring clear start-up situations is a great design pattern.

In the next step, various parameters must be written to the sensor in order to inform it about the way we want our data provided:

```
  sensor.setMeasurementMode(TEMP_AND_HUMID);
  sensor.setRate(ONE_HZ);
  sensor.setTempRes(FOURTEEN_BIT);
  sensor.setHumidRes(FOURTEEN_BIT);
  sensor.triggerMeasurement();
```

After the configuration is complete, it is time for harvesting the actual values. As Texas Instruments' example driver takes care of the conversion for us, the actual information collection can be done like this:

```
  float temperature = sensor.readTemp();
  temperature = sensor.readTemp();
  float humidity = sensor.readHumidity();
  printf("Temp: %f | Humi: %f", temperature,
 humidity);
  return 0;
}
```

We invoke the `readTemp` method twice for a reason. The HDC2010 family has a small oddity: after being configured for the first time, the temperature sensor always returns a value of -40°C. This value gets discarded immediately after the first read operation; by invoking the method twice, we can ensure that valid values are available to our program.

At this point, another invocation of *GCC* is needed. This time, both the driver class and the main file need to be parsed in order to prevent linker errors:

```
pi@raspberrypi:~ $ gcc HDC2010.cpp test.cpp
-lwiringPi
```

When done, temperature and humidity values can be harvested from the command line:

```
pi@raspberrypi:~ $ ./a.out
Temp: 21.170044 | Humi: 45.507812
```

Should you not believe our claims about the issue with the -40°C reset, feel free to remove the second invocation of the reading function. After ordering another recompilation, the temperature value returned will no longer be valid:

```
pi@raspberrypi:~ $ ./a.out
Temp: -40.000000 | Humi: 45.782471
```

What now?

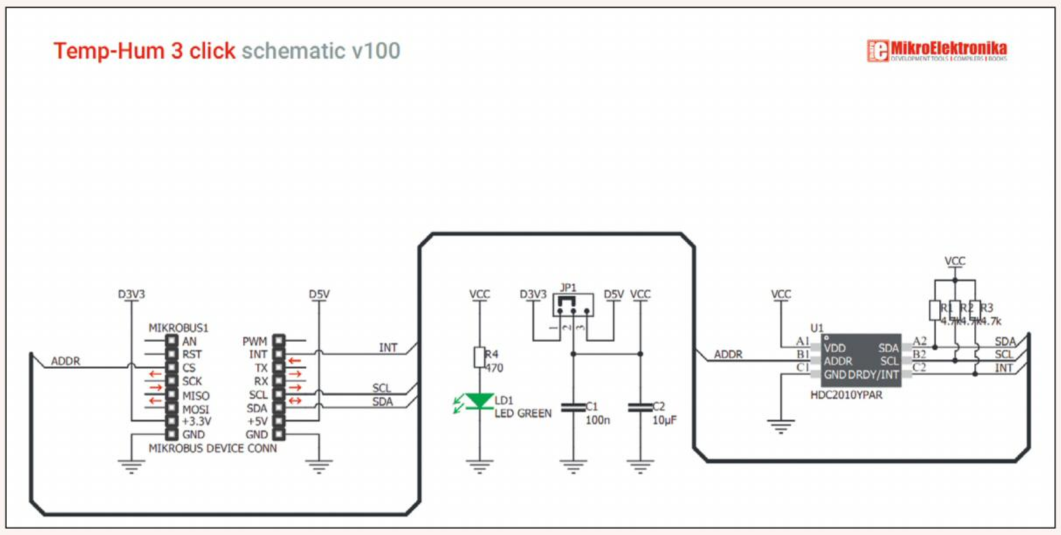
Our Raspberry Pi is able to collect both temperature and humidity information. It can be shared across the internet in a variety of ways – in our case, we used a simple proprietary protocol based on the Berkeley Socket API.

This, however, is a topic better left to software engineers. For the electrical engineering team, another interesting problem arises: given that I2C is intended to cover multiple devices, we should be able to add multiple HDC2010 instances. This raises a set of new questions, the answers to which we will introduce in another issue.

For now, we hope that you enjoyed our little experiments with the world of sensor interfacing on the Raspberry Pi. LXF

QUICK TIP

Should you, for some reason, feel uncomfortable using the wiringPi library, you can also fall back to the I2C library contained in the Linux kernel. More information on this programming approach can be found at http://bit.ly/lxf252i2c.



MikroElektronika's adapter PCB can work off both 3V3 and 5V. As the board does not, however, contain any active voltage regulators, there is no need to move jumpers around.

RASPIVID

Stream live to YouTube with your Pi camera

Christian Cawley explores how to create a YouTube livestream from your living room, with a Raspberry Pi and Camera Module.

OUR EXPERT

Christian Cawley has more Pis than you've had hot dinners, probably.

Streaming videos on the web is increasingly popular, and YouTube is the chosen platform for many. Almost any device can stream video to YouTube, but what if you want a dedicated streaming solution? The Raspberry Pi is a great choice for this, giving you the option to stream anything you like, just as long as there is an unlimited network connection.

You're limited only by your imagination, with the Pi and its camera module ideal for everything from a YouTube-based baby monitor to running live presentations (*see the box over there for more ideas*).

With a good network connection, the right Raspberry Pi set up, and a YouTube channel stream URL, you can have your Raspberry Pi YouTube streaming video camera up and running in just a few minutes.

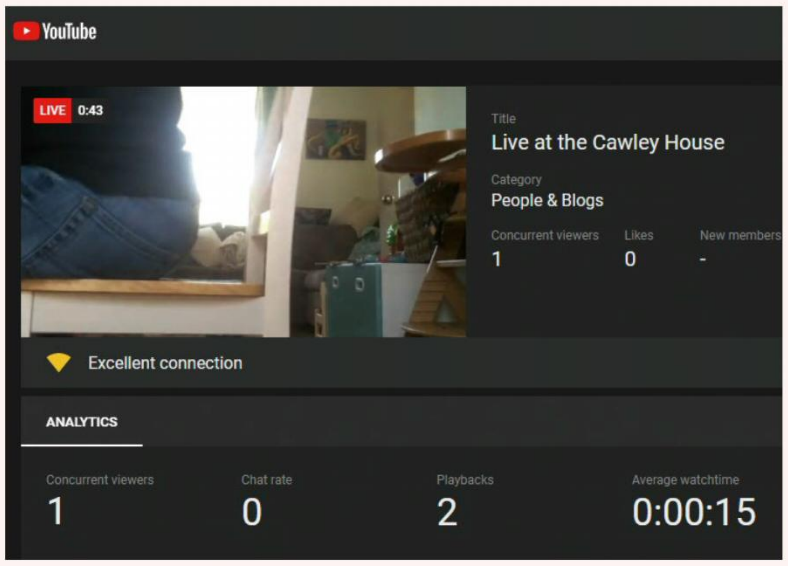
Streaming essentials

Before you even pick up your Raspberry Pi, it's a good idea to ensure your network is up to the task. The optimum bitrate for YouTube videos from the Pi is 400-600kbps. You can change this if necessary, but your router will need to be up to the job. Help things along by moving the Pi closer to the router, using an Ethernet cable if necessary, or employing powerline adaptors to improve signal strength at your Raspberry Pi's location.

While you don't necessarily need a wireless Pi, the best results can be enjoyed with a Raspberry Pi 3 or 3B+. The improved spec of these two over the Raspberry Pi 2 results in superior streaming performance. The Pi 2 will work, but you may run into resource bottlenecks. Tweaking that bitrate (you'll find out how later) will help if you're stuck using one.

Your choice of camera may impact how well this works. USB cameras, while compatible with the Pi, are largely pointless as the computer has such a good official camera. We used the Raspberry Pi Camera Module v2.1 for this project, but the older models should work just as well.

If you're planning to add audio to your camera stream, you'll also need a microphone, of course. The best option here is to use a Raspberry Pi-compatible USB mic, although anything connected to a suitable HAT will work. Don't use the TRRS port, however, as it won't work. Raspberry Pi audio is beyond the scope of this tutorial, so we're continuing with a silent video.



Make sure you have a clear picture and discernible audience for your YouTube stream. 'Man sitting on dining chair' is perhaps a niche too far.

We have a stream

The Raspberry Pi Camera Module must be connected before you power up the computer. If you haven't done this before, find the camera port and lift the catch. Insertion is straightforward: the rule of thumb to remember is that the silver side of the contacts point towards the HDMI port. Before booting up, it's also a good idea to connect the microphone for configuration later on.

With the Raspberry Pi up and running you'll need to configure the camera module. You can do this in the Raspberry Pi Configuration screen in the desktop environment, or via the *raspi-config* tool in the command line.

```
sudo raspi-config
```

Whichever method you choose, you need to display the Interfacing Options and set the Camera to Enabled. Select OK to confirm and reboot the Raspberry Pi when prompted. Next, you'll need to confirm that the camera is working. Use the *raspistill* command in the terminal:

```
raspistill –o image.jpg
```

This creates a file called **image.jpg** in the **/home** directory. You can check it has been created by listing the directory contents:

```
ls
```

In most cases, the camera image will be fine. To check it properly, you'll need to plug your Pi into a HDMI display, or check via VNC.

For streaming to work, you need a YouTube account and a unique key that is created for each stream. If you have a Google account you probably already have a YouTube account, so sign into YouTube and find the camera icon, click it, and select Go live.

In the New stream window, give the feed a title, set a Public/Unlisted/Private option, a description and genre, then click Create Stream. The Stream Preview window appears, displaying the standard Stream URL and the Stream name/key. You'll need this, so click the visibility icon and make a note of the key for later.

Streaming specifics

To create a video stream you should have the *avconv* software installed. This comes pre-installed in Raspbian Stretch, so if you're using an older version it's wise to update first.

```
sudo apt upgrade
sudo apt update
```

If you don't want to do this (perhaps because you've got deprecated software that you don't want to stop using), simply install *avconv* as part of the libav-tools package instead:

```
sudo apt install libav-tools
```

Creating the stream requires the input of a long `raspivid` command, consisting of various switches and instructions such as `fps` to set the frames per second. As it's quite a complicated command, it's worth looking at some of its key elements.

`-fps` As noted, this sets the frames per second captured by the camera. Anything over 24fps should be fine; less than this and it may resemble time-lapse. A lower rate can improve streaming, however.

`-w <xxx> -h <xxx>` Specifies the width and height of the video. Without these, *raspivid* defaults to 1920x1080 HD resolution.

`-b` The output bitrate limit, which YouTube recommends should be 400-600Kbps. A lower figure means a lower-quality video, but better streaming.

`-acodec` If you're not using a mic you'll need to include this. YouTube doesn't permit videos without audio (or audio without video), so this command creates a fake audio track. If you have a mic connected and working, you can omit this.

`-f` The output format. For streaming video from YouTube to the Raspberry Pi, FLV is the best option.

Keep on streamin'

If you're setting this up over SSH it's a good idea to take steps to avoid the stream closing when you disconnect. One way of avoiding this is with *screen*.

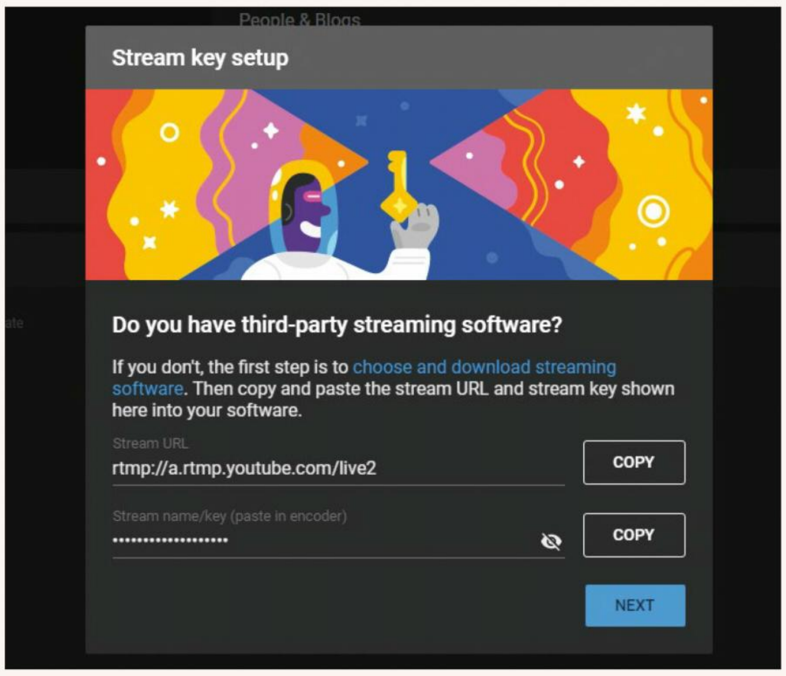
```
sudo apt install screen
```

Wait while it installs, then reboot the Pi.

```
sudo reboot
```

Connect again over SSH, then enter the `screen` command. You can then run the `raspivid` command. Note that this should be entered as a single line, like so:

```
raspivid -o - -t 0 -vf -hf -fps 30 -b 6000000 | avconv -re
-ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i /dev/
zero -f h264 -i - -vcodec copy -acodec aac -ab 128k -g 50
-strict experimental -f flv rtmp://a.rtmp.youtube.com/
live2/[your-secret-key-here]
```



Copy the key for your RMTP stream into the raspivid command to stream footage from your Raspberry Pi to YouTube.

Simply copy the YouTube stream key from the browser window into the terminal, replacing `[your-secret-key-here]`. Press Return and wait while the stream starts. A moment later, you'll see it running in the top-left corner of the YouTube livestreaming page. If nothing happens, double-check all the setup steps as listed above.

Crossing the streams

With the stream running, you can share it or simply watch it. Using a private channel for the feed at the setup stage lets you limit the feed to YouTube apps or browser sessions with the corresponding account details. Otherwise, the feed will be public or unlisted.

With both options, you can share the feed with others. The public feed is open to anyone, whereas the unlisted feed is limited to those who have the feed URL; it won't be displayed in search results. Both options let you share the feed URL on your preferred social networks as well. LXF

QUICK TIP

Want to livestream your life to YouTube via your Raspberry Pi? Try a high-capacity rechargeable portable battery, with enough charge to last a day. Beware: they can get heavy!

» USES FOR LIVESTREAMING

Why might you want to stream video with a Raspberry Pi? Various uses can be considered, although some will be impacted by latency, which can be as much as 30 seconds once the video has been captured, sent and uploaded to YouTube.

For example, you could set up the Raspberry Pi as a baby monitor. Configuring the YouTube feed as private means that you can keep an eye on your little ones from any portable device, just as long as you're signed in with the same account.

Alternatively, you might use the Raspberry Pi as a security camera. Again, a private channel can let you watch your home from any location you can get a mobile signal. While it will miss the benefits of a standard IPTV security camera (such as motion detection), it makes an adequate substitute.

Other problems can be solved with a Raspberry Pi streaming video camera. You could go public with your YouTube video feed, perhaps making short livestreams to share your thoughts about recent topical events. You could demonstrate your job, your work setup, or even a hobby you enjoy; you might even conduct an AMA with your Raspberry Pi as the stream host, direct to YouTube.

MCFLY

McFly: Get more out of your Bash history

While **Shashank Sharma** swears by the powers of Bash's history expansion, there's another tool that can help you to be more efficient.

OUR EXPERT

Shashank Sharma is a trial lawyer in New Delhi and avid Arch user. He's always on the hunt for budget-friendly geeky memorabilia.

Open source leads to freedom. Freedom leads to choice. Choice leads to angry, neverending *Vi* vs *Emacs* and similar discussions on the internet. This paraphrasing of Yoda's famous quote sums up the open source community rather well – but it also leads in to our tutorial.

For anyone who's spent time on the terminal, the history expansion feature in *Bash* is unparalleled in its usefulness. Just as brilliant is the reverse search, which you can invoke with Ctrl+R. Reverse search lets you search for the specified keyword in *Bash* history, so that you don't have to type in the same command over and over again, saving some precious time and effort.

Written in Rust and released under the MIT licence, *McFly* is a replacement of the default Ctrl+R *Bash* history search. While it performs the exact same function as the traditional history search, *McFly* boasts an intelligent search engine powering its functions, which results in smarter suggestions.

The default Ctrl+R merely provides a list of commands that match your search criteria, without context. In comparison, when suggesting commands from your history, *McFly* considers the directory from which you last ran the command, believing that you're likely to run the same command from the same directory again. It also considers the frequency, or how often you run that command, the context – the commands that were executed immediately before and after the suggested command – and even the exit status of the command. This is especially helpful because no one wants to keep running commands which only fail to execute for one reason or another.

McFly is able to provide these robust features thanks to its underlying neural network, which runs in real time. Despite the extensive 'thinking' that it does prior to making suggestions, *McFly* is just as fast as a Ctrl+R look-up. Best of all, the project maintains an unmolested *Bash* history, which means you can stop using *McFly* whenever you want.

As with most projects hosted on GitHub, the

» UNDERSTANDING HOMEBREW

Unlike other package managers, the advantage with Homebrew is that it installs packages to your **/home** directory, thus doing away with the need to use *sudo* when installing new packages.

If you already have *git*, *curl* and other developmental tools installed on your machine, you can install Homebrew using the official script:

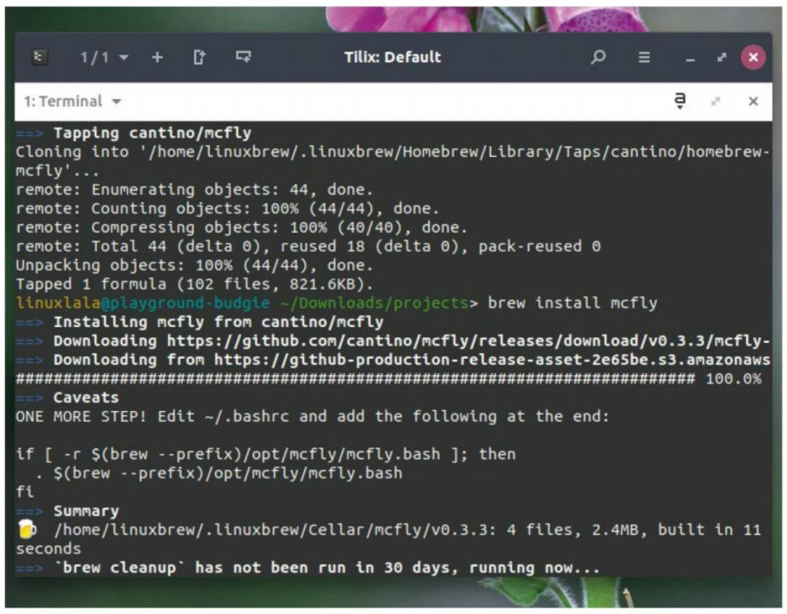
```
$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.sh)"
```

This command downloads the shell script using *curl*, and then executes it (confusingly, Homebrew on Linux used to be called *Linuxbrew*). You'll be prompted to provide the *sudo* password, which enables the script to install Homebrew in the **/home/linuxbrew/.linuxbrew** directory. You can alternatively install it within the current user's **~/.linuxbrew** directory.

The next step is to add certain directories to your $PATH so as to be able to use Homebrew:

```
$ echo 'export PATH="/home/linuxbrew/.linuxbrew/bin:/home/linuxbrew/.linuxbrew/sbin/:$PATH"' >>~/.bashrc
$ echo 'export MANPATH="/home/linuxbrew/.linuxbrew/share/man:$MANPATH"' >>~/.bashrc
$ echo 'export INFOPATH="/home/linuxbrew/.linuxbrew/share/info:$INFOPATH"' >>~/.bashrc
```

Finally, source the **~/.bashrc** file for the changes to take effect with the `source ~/.bashrc` command.



You'll find packages installed via Homebrew in the /home/linuxbrew/.linuxbrew/bin directory.

installation is fairly straightforward. *McFly* provides detailed instructions on its site, and recommends using Homebrew to install it. If you're new to this, refer to the 'Understanding Homebrew' box on page 58.

To install *McFly*, open a terminal window and run the following commands:

```
$ brew tap cantino/mcfly https://github.com/cantino/mcfly
==> Tapping cantino/mcfly
Cloning into '/home/linuxbrew/.linuxbrew/Homebrew/Library/Taps/cantino/homebrew-mcfly'...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 44 (delta 0), reused 18 (delta 0), pack-reused 0
Unpacking objects: 100% (44/44), done.
Tapped 1 formula (102 files, 821.6KB).
```

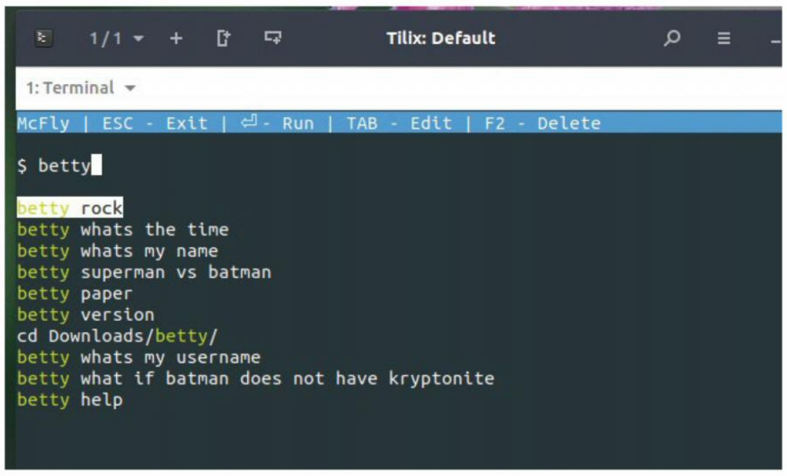
This command merely clones the *McFly* repository, not unlike the `git clone` command, but it does so within the Homebrew hierarchy. We still need to install *McFly*, however, and you can do that with the `brew install mcfly` command:

```
$ brew install mcfly
==> Installing mcfly from cantino/mcfly
==> Downloading https://github.com/cantino/mcfly/releases/download/v0.3.3/mcfly-
==> Downloading from https://github-production-release-asset-2e65be.s3.amazonaws
######################################## 100.0%
==> Caveats
ONE MORE STEP! Edit ~/.bashrc and add the following at the end:
if [ -r $(brew --prefix)/opt/mcfly/mcfly.bash ]; then
 . $(brew --prefix)/opt/mcfly/mcfly.bash
fi
==> Summary
🍺 /home/linuxbrew/.linuxbrew/Cellar/mcfly/v0.3.3: 4 files, 2.4MB, built in 11 seconds
==> `brew cleanup` has not been run in 30 days, running now...
```

As you can see from the output, you're instructed to edit the **~/.bashrc** file and add a couple of lines to it. You can do this using your favourite editor, be it *Vim*, *Emacs* or the unassuming *Nano*. With that done, you must run the `source ~/.bashrc` command so that the changes take effect. With the installation out of the way, you can now use *McFly* as you would Ctrl+R on *Bash*.

When you run *McFly* for the first time with the `mcfly` command, the tool will import your history file, and other pertinent information such as execution status, time stamps and so on into a SQLite database. Depending on the size of the history, this process can take several minutes. You can start using *McFly* to perform searches as soon as that process is completed.

The tool supports various subcommands such as `add` and `search`. Use the former if you wish to add a command to the history. For the most part, you'll be using *McFly* to search for commands, not unlike the Ctrl+R reverse search. Unlike that search, which operates at the command prompt itself, *McFly* instead



You can switch between McFly and the traditional reverse search by commenting out the 'if' code block added into the ~/.bashrc file.

takes over the entire terminal window. The `search` subcommand expects a keyword, which it matches through the database and then presents you with relevant results.

The *McFly* interface features a list of possible actions at the top. You can use the up and down arrows to scroll through the list of suggested commands. The selected command is executed when you hit Return. If you want to edit a command before executing it, you can do that by selecting it from the list and pressing Tab. You're dropped back to the command prompt, with the selected command typed in ready for your edits.

You can also use the *McFly* interface to delete a command from the history by selecting it from the list and pressing F2. If none of the suggested commands interest you, press Esc to return to the command prompt instead.

Thankfully, you don't have to type `mcfly search <keyword>` every time you wish to perform a reverse look-up. The project replaces the existing Ctrl+R reverse search implementation, so pressing Ctrl+R also launched the *McFly* interface.

Advanced usage

Apart from `search` and `add`, *McFly* also offers additional subcommands. Since the tool keeps a record of the directory from which certain commands are executed so as to provide relevant recommendations, you can use the `move` subcommand to update the database if you move a directory.

For instance, if you store all your quirky GitHub projects in the **~/Downloads/projects** directory, but want to move the **projects** directory to the **home** directory, you can update the *McFly* database with `mcfly move <old-dir-path> <new-dir-path>`:

```
$ mcfly move ~/Downloads/projects ~/projects
McFly: Command database paths renamed from /home/linuxlala/Downloads/projects to /home/linuxlala/projects (affected 8 commands)
```

As you can see, *McFly* automatically updates the database, which is stored in the **~/.mcfly/history.db** file, and informs you of the total number of affected commands.

Although it's basically pretty straightforward, the project might seem like overkill for regular home users who only use the terminal irregularly, and don't have enough commands polluting their *Bash* history. Which is fair enough – but if you're a *Bash* addict, *McFly* will soon become your best friend! **LXF**

QUICK TIP

The added advantage of installing McFly with Brew is that it makes uninstallation a breeze. All you need to do is run the 'brew uninstall mcfly' command. You can then remove the repository as well with the 'brew untap cantino/mcfly' command. Finally, edit the ~/.bashrc file and remove the lines of code that you were instructed to add post-installation.

Credit: https://calibre-ebook.com

CALIBRE

Manage your ebook collection efficiently

Nick Peers immerses himself in ebooks with a guide to viewing, managing, converting, editing and even streaming them.

OUR EXPERT

Nick Peers has amassed quite an ebook collection over the past 24 months – all to do with WW2.

QUICK TIP

***Calibre* can also be used to catalogue your paper books: choose 'Add books by ISBN' to input books by ISBN number – *Calibre* will then download metadata and covers for them.**

Fed up with lugging paperback books around with you? Running out of shelf space? The solution – of course – is to digitise your library. When it comes to collecting, managing and distributing ebooks to your various ebook reading devices, there's no better tool than *Calibre*, an open source, one-stop-shop for all things ebook-related, from viewing, cataloguing and organising to converting, sharing and even editing. We've got a lot of ground to cover, so let's crack on.

Avoid installing *Calibre* through the Ubuntu Software Centre – it's incredibly outdated; instead, open a terminal window and issue the following command:

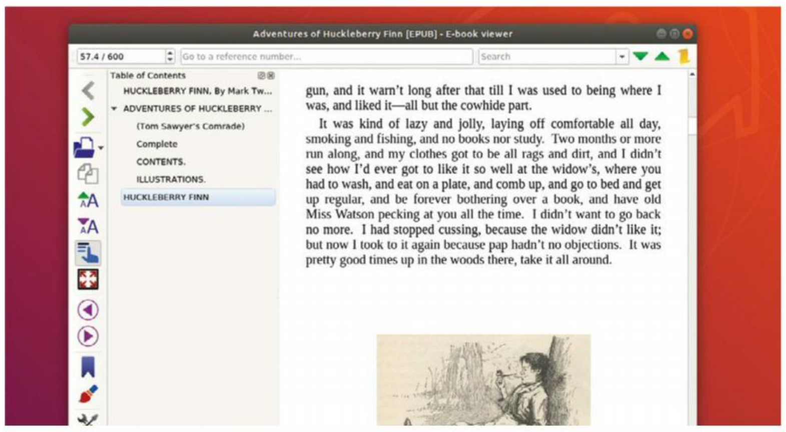
```
$ sudo -v && wget -nv -O- https://download.calibre-ebook.com/linux-installer.sh | sudo sh /dev/stdin
```

After installation, you'll be able to launch *Calibre* through your launcher as well as via the terminal. You'll start with the welcome wizard – set your language to English (UK), choose where to store your ebook library (all books you import from other destinations will be copied here) and click Next.

You'll next be prompted to select any ebook devices you own – these can be dedicated ebook readers or a smartphone or tablet. Be sure to select your exact make and model – *Calibre* will automatically convert ebooks into compatible formats when you transfer books to your reader. If it's not present – or you don't have one – select Generic and click Next to complete the wizard.

You'll find yourself at the main *Calibre* library screen – as time goes on, this will fill with all your added ebooks, but for now just one solitary entry is present: a quick-start guide, which you can view in *Calibre*'s built-

Click the Layout button in the bottom-right corner to give Calibre's book list a more attractive look and feel.



Calibre comes with a built-in ebook viewer that can be configured how you like it – there's also a full-screen mode available.

in ebook viewer by double-clicking it. A new window will open, with all the controls you need to read, navigate around and search your book.

The buttons down the left-hand side are straightforward enough. The < and > buttons are back and forward navigation buttons like a web browser; instead, use the purple arrows further down to move between pages. You'll also see options for copying selected text to the clipboard and tweaking the font size for readability purposes.

The ebook viewer remembers where you left off in a book, so you'll jump straight back to the point where you left it – but you can also insert bookmarks, which are embedded into EPUB files. If the book is then shared with a friend, they'll have access to those bookmarks. There's also a Reference mode to help you find your place in the book; when switched on you can highlight individual paragraphs and use the section and paragraph numbers to help determine where you are. Share these with others or use them to jump straight back in using the 'Go to a reference' box at the top.

The ebook viewer also boasts search tools and there's a Preferences button to enable you to customise the layout (including fonts, colours and backgrounds) to your tastes. You can also save different setups as themes, enabling you to switch between them easily. Advanced users should check out the 'User stylesheet' tab, where you can apply a consistent look across your ebooks with the help of CSS. The 'here' link points to an online thread providing examples for you to experiment with.

There are several ways you can add new books to your *Calibre* library. Clicking 'Add Books' enables you to add individual books one at a time – each book is copied into your *Calibre* library folder, organised into sub-folders by Author\Title. Don't like this? Like so much in *Calibre*, you can choose a different organisation structure; in this case, open Preferences and select 'Saving books to disc' to pick a different folder template to use.

If you have a collection of books in a single folder or zip archive, save time by clicking the down arrow next to 'Add Books' and select the appropriate 'Add books...' option depending on how your books are currently organised. Once added, books appear in the main window in list format, sorted by 'date added' by default. Selecting one displays its cover, author, IDs, tags, format(s) and shortcut to the folder path in the right-hand pane, allowing you to quickly open the folder containing the file. Click the author's name, and you'll be whisked to Goodreads.com to read a short biography and view other titles they've written. If filled, the IDs field whisks you to a web link for the book itself.

Once your collection has grown to a certain size, you'll want ways of organising it, and this is where tags come into their own. Click a tag and you'll see that the Search box is populated with the tag in question, and the list is filtered to only display books containing that specific tag. The Search box can also be used as a regular keyword search tool, or click the gear icon for more specific searches involving exact matches as well as looser AND, OR and NOT-style searches.

Searches can be saved as so-called 'virtual libraries' – a clever way to group related content together without tying books to single collections. After you've perfected your search terms, click 'Virtual library' and choose 'Create Virtual library' to save it. Once created, you'll see an option to display your virtual libraries as a series of tabs, making it quick and easy to move between them.

Edit book metadata

Your organisational structure relies on the metadata stored with each book. It's unlikely that all the books you import will adhere to a single standard, which is where *Calibre*'s metadata-editing tools come to the rescue. Select one of your books and click the 'Edit metadata' button where you'll be able to manually change (or add) various bits of information from author and title to cover artwork and tags.

Manually adding this information can be a chore, particularly if it's blank or incorrect, but there are some time-saving tools included. All you need to do is supply a title, author or ISBN (enter this into the ID tag), then make use of the automated 'Download metadata' button, which will search the web for matches and provide you with metadata and potentially a cover too. If more than one match is found, check all available options, then select the correct match or edition for your book before clicking OK.

If there's no available cover, click 'Generate cover' to generate a generic cover using the given title and author. Keep clicking the button to cycle through different styles or click and hold to bring up the 'Customize the styles and colours of the generated cover' dialogue box. Once you're happy with your changes, click OK to close the box, or click Next to move on to the next book in your collection. Be sure to click OK at some point to commit your changes – note these are written to the *Calibre* library only, not directly embedded in the files.

Click the down arrow next to 'Edit metadata' and you'll reveal several other time-saving tools: an option to quickly download metadata and covers for a book, for example, plus the option to edit metadata in bulk – you'll need to select your books first with the usual Shift+click and Ctrl+click combos. This reveals two tabs: 'Basic metadata' is the best option when attempting to introduce some order into your library, as you can use this to easily add tags to multiple titles at once. There's also a 'Search and replace' tab, which should be used with caution.

Calibre supports a wide range of ebook formats, from plain text to PDF as well as the industry standards, such as ePub, HTML, MOBI and AZW. If you're using *Calibre* exclusively as a tool for organising and reading ebooks, that's all you need to know. If, however, you're planning to edit an ebook or send it to a specific device, you may run into problems that require you to make use of *Calibre*'s powerful conversion tool.

For example, you can only edit ebooks in the EPUB or AZW formats, while certain devices only support specific formats. To resolve this, select one or more troublesome books and choose 'Convert books'. Start by selecting your chosen output format from the drop-down menu: EPUB for non-Kindle readers and AZW3 for Kindle. You'll also see a range of options down the left-hand side of the window enabling you to tweak the

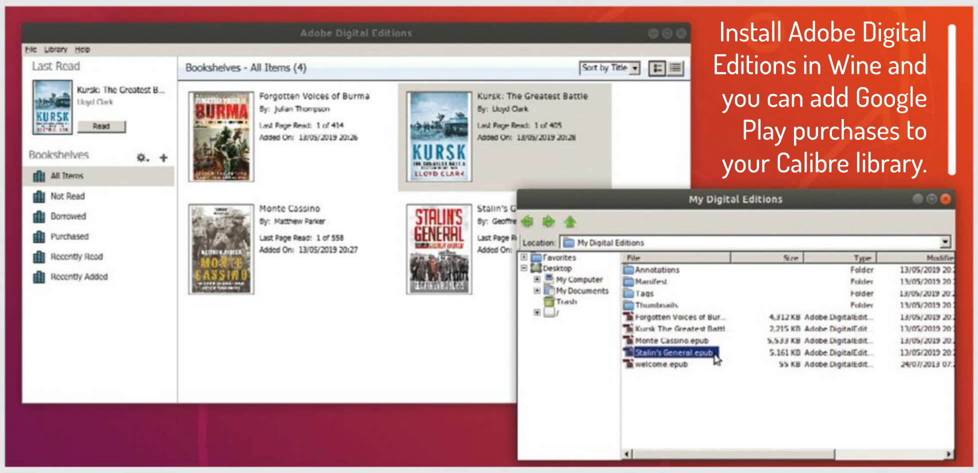
QUICK TIP

***Calibre* provides its own DRM-free catalogue of ebooks for download (and in some cases purchase). The quickest way to access this (and other ebook downloads, both free and paid-for) is through the program itself: click Get Books to open a search engine. Free sites to focus searches on include Project Gutenberg and Archive.org.**

» CALIBRE AND DRM

While *Calibre* can only open DRM-free books with its own ebook reader, you can still add Kindle content to your library (see **http://bit.ly/lxf252drm**). From here you can transfer it to a Kindle app or device, but you can't open it directly in *Calibre* – at least not without the help of a third-party plug-in. It's called DRM Removal Tools for eBooks, which you can download and install following the instructions at **http://bit.ly/lxf252drm2**. Note that breaking DRM is illegal in many territories, even if you've purchased the original and simply want to remove any restrictions on what device you can read it on.

Some DRM-protected ebooks – including purchases from Google Play Books – require *Adobe Digital Editions*, which isn't available natively. We managed to get it installed and working following the instructions at Pat David's blog (**http://bit.ly/ade-linux**) along with installing *winbind* (`sudo apt-get install winbind`). You can then download EPUBs from **https://play.google.com/books**. Once done, you'll be able to add these books to your Calibre library (you'll need the DRM Removal Tool to open them).



Install Adobe Digital Editions in Wine and you can add Google Play purchases to your Calibre library.

> **QUICK TIP**
> When editing ebooks, *Calibre* offers a handy 'checkpoints' function, which enables you to experiment safely in the knowledge that you can roll back to an earlier version easily. Before making a major change, choose 'Edit > Create checkpoint'. Give it a name and click OK. Use the Edit > Revert... options to move between checkpoints.

book's layout and content. Ebook conversion isn't an exact science, so start by simply checking that the Page Setup tab is formatted for your target ebook reader if applicable. Click OK and a new copy of the file is generated; you'll be directed towards the bottom right of the main *Calibre* window and a Jobs: status indicating the conversion is being performed. When it reverts to 0, the job is complete.

You'll now see that two different versions of the same book are listed in its Details pane – click the new format to view it. If you're unhappy with the conversion, click 'Convert books' again and experiment with new settings to try to correct any issues. When you click OK a new copy is generated, overwriting the old one.

One of *Calibre*'s strengths is its wide support for ebook readers, from the Kindle to your Android phone or tablet. Most ebooks readers should automatically appear when connected to your PC, but generic devices like an Android tablet may need more configuration.

If you still can't get it recognised after switching on USB debugging and setting USB configuration mode to MTP (Media Transfer Protocol), try going down the wireless route: install the *Calibre Companion* app – it costs £2.99, but a free demo version is available. After installing, click 'Connect/Share > Start wireless device connection'. Leave the default settings as they are and click OK. Now open the app and click 'Connect > As Wireless Device'. Note you'll also need a separate app for viewing ebooks – the *Kindle* app will work with MOBI and AZW files, otherwise try *Moon+* or *FB Reader*.

Once connected, you'll see new buttons appear in the main *Calibre* window: Send to Device, Library and Device. Use the latter two buttons to switch views, and the 'Send to Device' button to quickly transfer items to your ebook reader. Remember to convert your books first if necessary – EPUB for Android, MOBI for Kindles. The connection is two-way – select Device to view all books currently on the device, then right-click one and choose 'Add books to library' to download it.

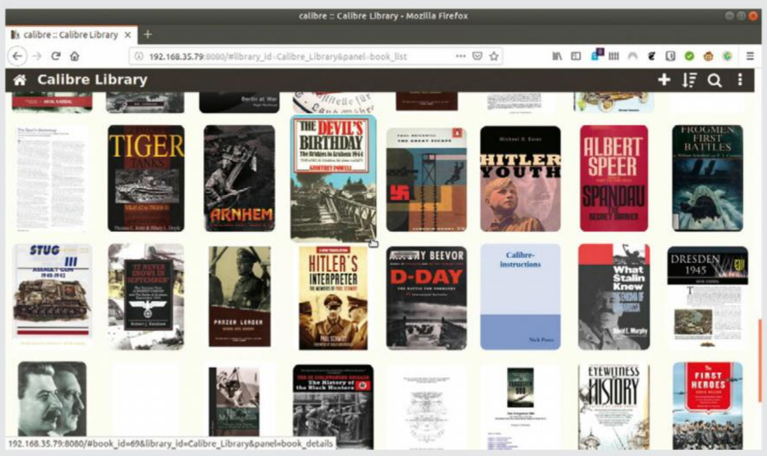
Another way to use *Calibre* to serve content wirelessly is by making use of its server capabilities – see the box *(below left)* for more details.

» CREATE YOUR OWN EBOOK SERVER

Why go to the trouble of manually copying ebooks to all your wireless devices when you can stream them on-demand over your local network? *Calibre* has a built-in content server that, when enabled, can be accessed through any web browser – either on another computer or through your mobile device. It's incredibly simple to use: simply click 'Connect/share' and choose 'Start Content server'. Then click 'Connect/share' again and you'll see the IP address plus the port number (8080 by default) you need to type into your remote web browser. You can also browse the library on Android devices through the *Calibre Companion* app.

The web interface is simplified and optimised for touch support – you'll see a list of thumbnails you can tap to view a summary, and access options for reading or downloading the ebook to your device. There are also various options for searching, sorting and managing your collection at the top of the screen.

The built-in reader automatically goes full-screen when selected and provides some simplified navigation controls – touch the screen, click at either edge or use the cursor keys to navigate. Click back on the browser to return to the book summary page.



Once it's set up, you can browse your Calibre library from other PCs or devices connected to your home network.

Edit and create ebooks

You can also use *Calibre* to edit existing ebooks or create new ones from scratch. The step-by-step guide *(right)* reveals how to get started with a book from scratch – the building blocks of ebooks are effectively HTML files for text and formatting, and embedded images, which can then be placed in the text itself. One tip: if you don't like the cover, design your own, import it as an image and then right-click the file under Images and choose 'Mark filename.jpg as cover image'

Another essential step in creating your own ebooks is to insert a table of contents to help readers navigate your books. This is where applying header styles to chapter and sub-chapter titles pays dividends, as all you need to do is choose 'Tools > Table of Contents > Edit Table of Contents'.

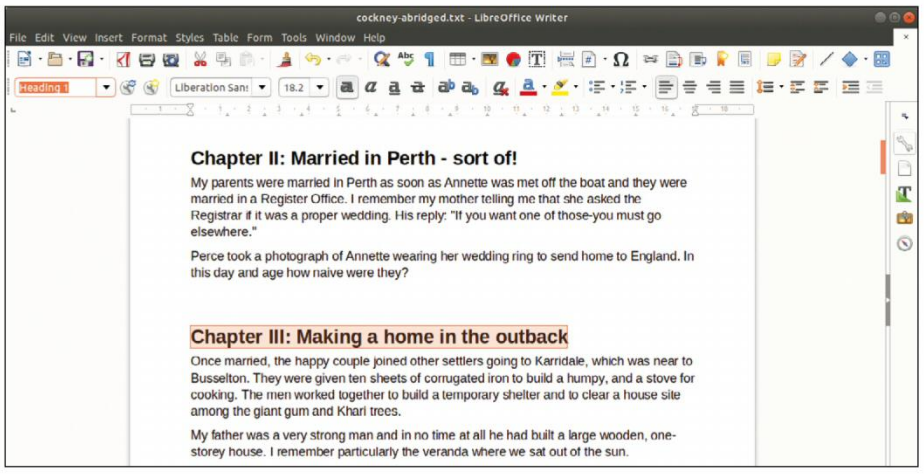
From here, you'll see options for automatically inserting entries using 'major headings' (text marked 'Heading 1' only) as well as all headings. You can also generate other types of entries from inline links, as well as manually input your own. Once created, choose 'Insert inline Table of Contents' from the same 'Tools > Table of Contents' menu to insert the table of contents into your book. It's also possible to edit existing ebooks, but only if they're in AZW3 or EPUB format, so if necessary, convert them first as outlined earlier.

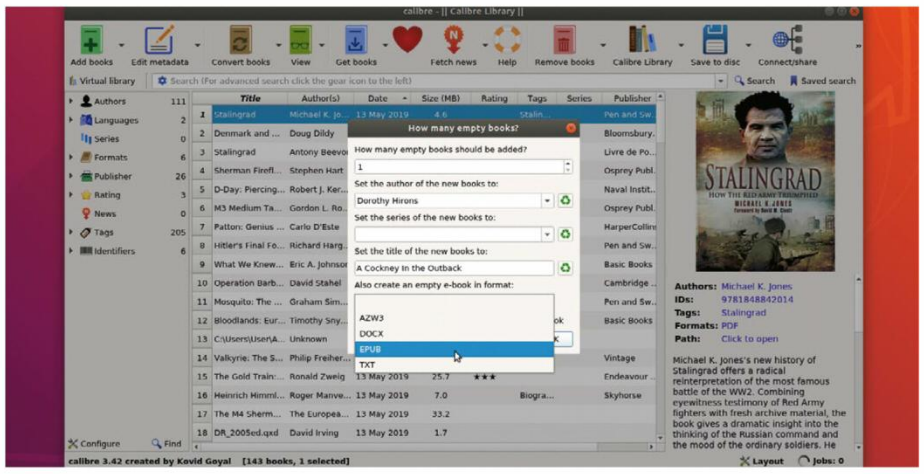
Calibre's ebook editor is basically a glorified HTML editor, so it helps to have a working knowledge of HTML to take your edits to the next level if you're looking to make changes to individual parts of the book. More likely, however, is your desire to apply consistent styles across your book.

Rather than wrestle with CSS and HTML, this part of the process is best achieved outside the editor, so save your work and close it. Next, choose 'Convert ebooks' to change other fundamental aspects of your book. Make sure the output format matches the input one, then explore the 'Look & feel' section for settings controlling fundamental aspects of your ebook's layout.

These range from embedding fonts into the document (tick 'Subset all embedded fonts' to keep the file size down) to whether or not text is justified and the spacing between paragraphs. The result is a professional-looking ebook that you can view on any device and distribute to friends. LXF

CREATE YOUR OWN EBOOK



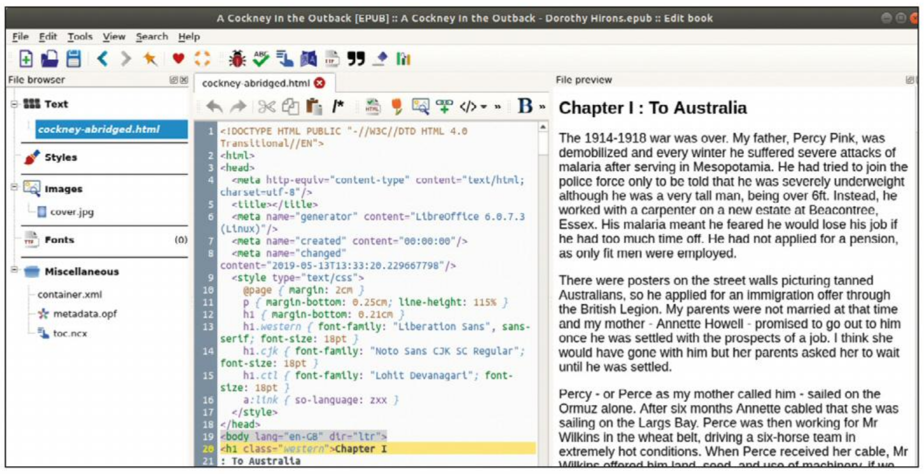


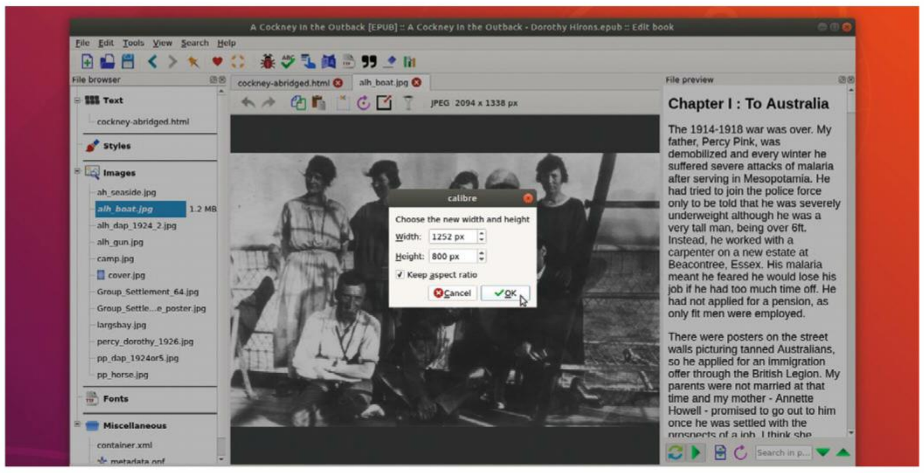
1 Prepare your text

Open your word processor and import the text of your book. First, you need to format the text using the built-in text formats: for example, select any chapter titles and format them Heading 1. You can then apply Heading 2 and other headings to sub-headings in the document – these will then appear in the table contents. When done, save the file in HTML format.

2 Create an empty ebook

Open *Calibre* and choose 'Add books > Add empty book (Book entry with no formats)'. Enter the author and title details, and opt to create an empty ebook in either EPUB or AZW3 format. Click OK and an empty book will be created. Click Edit Metadata to add additional information about your book, plus attach or generate a cover image.



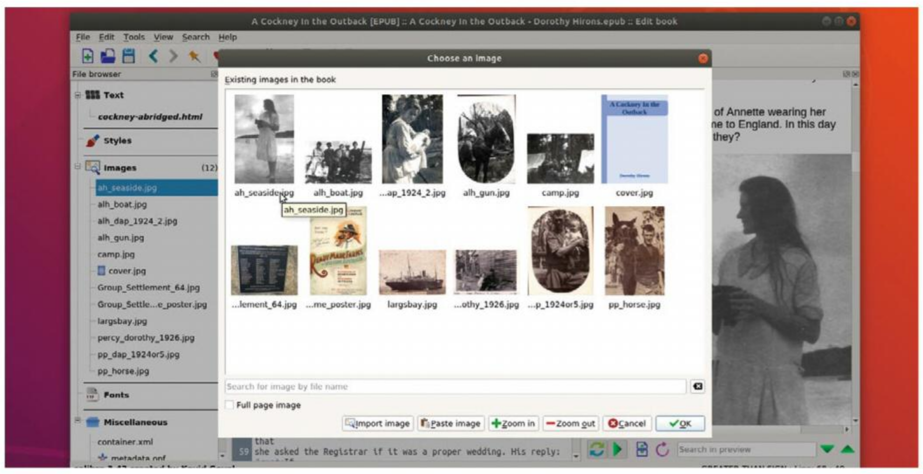


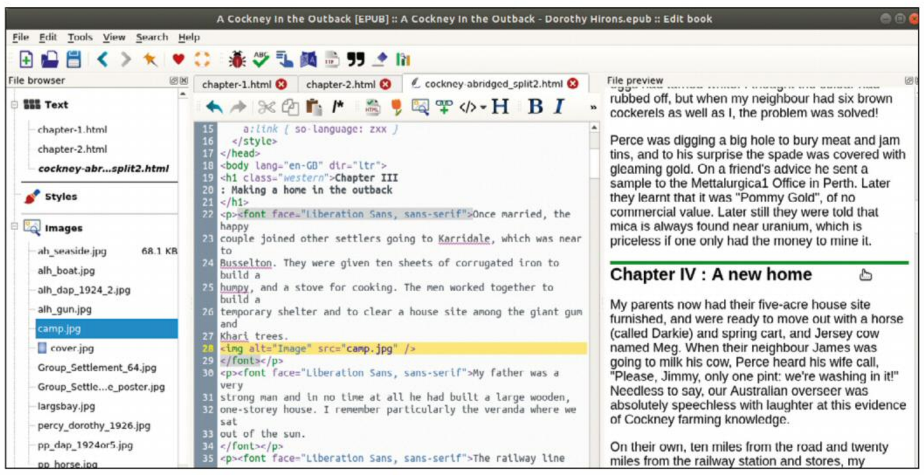
3 Import text file

Click 'Edit book' to open the main editing window. Right-click start.xhtml in the left-hand pane and choose 'Replace start.xhtml with file...' Select the file you created in step one. When inserted into the file, double-click it to edit the file – you can see how your book looks in the right-hand pane, but you'll need to edit the actual HTML to make changes.

4 Add images

Next, choose 'File > Import files into book' to import any images you plan to use. You'll see these appear in the Images section of the book. Double-click an image to make changes to it – you can resize an image to reduce its size, plus there are various filters that enable you to perform additional enhancements. Changes only affect the images in the book, not the original files.





5 Incorporate images into book

Place the cursor at the point in your HTML where you'd like to insert an image, then click the 'Insert image' button. Select one of the images in your book and it'll immediately appear in the text. If it's too large, remove the reference, edit the photo as per the previous step and insert it again. Those with degrees in HTML coding can go further and align the pictures to the text.

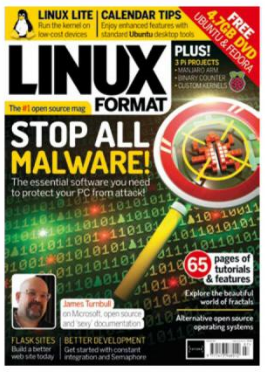
6 Split by chapter

Introduce page breaks between chapters by splitting your HTML file at a chapter point. Click the split file button underneath the preview window and then use the green divider to choose where to introduce the break. Once done, click and the file will split in two. Right-click the first HTML file and choose Rename... to give it a more descriptive name such as 'Chapter 1'.

BACK ISSUES » MISSED ONE?

ISSUE 251
July 2019

Product code:
LXFDB0251



In the magazine
Stop all Linux malware! Yes, we really mean it – just follow our guide. Create a database-driven website, discover open operating systems in our *Roundup*, build your own custom Raspberry Pi image and meet a Microsoft employee. (Be nice.)

DVD highlights
Ubuntu 19.04 "Disco Dingo" (64-bit) plus Fedora 30 (64-bit).

ISSUE 250
June 2019

Product code:
LXFDB0250

In the magazine
Get Linux on your Chromebook with our complete install guide. Plus check out our Roundup of rescue distros, tweak your Pi config to perfection, optimise *Steam* or ditch it altogether with *Lutris*, and meet the fine folks behind 64 Studio.

DVD highlights
Solus Linux 4.0 (64-bit) and Sparky Linux 5.7 (32- and 64-bit).

ISSUE 249
May 2019

Product code:
LXFDB0249

In the magazine
Proprietary voice control systems be gone! Build your own open source smart home. Plus, keep track of your Git repos, edit audio in *Audacity*, learn WebAssembly with Rust, check out our *Roundup* of disk cloning tools, and create a stop-motion studio.

DVD highlights
Kali Linux 2019.01 64- & 32-bit: the penetration testing kit used by pros!

ISSUE 248
April 2019

Product code:
LXFDB0248

In the magazine
Find out how to relive the entirety of computing history (well, most of it) through open source. Build your own Pi Camera photobooth, check out our *Roundup* of vector graphic editors, build a better *WordPress* server, and make scanning easy.

DVD highlights
Qubes OS 4.0.1 64-bit, possibly the most secure Linux distro to date.

ISSUE 247
March 2019

Product code:
LXFDB0247

In the magazine
Lock down Linux Mint 19.1 tighter than the ed's wallet with our full guide to security. We take at look at the huge range of photo editors available in *Roundup*, show you how to get your head around Git, build a 2D platform game, and make your terminal talk.

DVD highlights
Linux Mint 19.1 Cinnamon 64-bit and Xfce 32-bit editions. Tasty!

ISSUE 246
February 2019

Product code:
LXFDB0246

In the magazine
We bring you 100 of the best open source tools, ranging from audio and video software, to file management and security, and more. There's tutorials on desktop publishing, *Emacs* and Middleman, plus how you can get into open source astronomy.

DVD highlights
It's a Linux starter pack: Bodhi Linux 5.0, Voyager 18.10 and Sparky 4.9!

To order, visit myfavouritemagazines.co.uk

Select Tech from the tabs of magazine categories, then select Linux Format.

Or call the back issues hotline on **0344 848 2852**
or **+44 344 848 2852** for overseas orders.

Quote the issue code shown above and have your credit or debit card details ready

UK subs turn to p24

NOT FROM THE UK?

SUBSCRIBE!

Don't wait for the latest issue to reach your local store – subscribe today and let *Linux Format* come straight to you!

3 GREAT WAYS TO SUBSCRIBE

Print, digital, and print & digital bundles!

» USA
From $15
every 3 months

» REST OF THE WORLD
From $15
every 3 months

» EUROPE
From €15
every 3 months

IT'S EASY TO SUBSCRIBE!

Click: **www.myfavouritemagazines.co.uk/sublin**

Call: **+44 344 848 2852**

Lines open 8am–7pm BST weekdays, 10am–2pm BST Saturdays*

Savings compared to buying 13 full-priced issues. You'll receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Your subscription is for the minimum term specified and will expire at the end of the current term. Payment is non-refundable after the 14-day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at time of print and subject to change. *UK calls will cost the same as other standard fixed-line numbers (starting 01 or 02) and are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit **http://bit.ly/magtandc**. Offer ends 31 August 2019.

FFMPEG

Convert GIFs to video

Tired of stuttering GIFs on your blogs and websites? **Dan Dart** explains how to turn your best memes into silent video files instead.

OUR EXPERT

Dan Dart
Polyglot from a young age, code fanatic and creator of video sharing project Project Chaplin and the web-based desktop environment Bibud, Dan is known as 'The Cloud Man'.

The GIF format is from another age: originally created in 1987, tweaked in 1989, it was designed for simple, blocky colours. While you may hear that it only supports 256 colours, that restriction has been 'hacked out', by adding as many palettes of 256 colours as needed to an image (often bloating it significantly), and then changing the palettes across the picture.

This can cause a lot of problems with size and loading times, with content deviating significantly from the original specification. Photos with millions of colours can often have a filesize, and thus a loading time, several orders of magnitude more than file formats actually designed for photography, such as JPEG, and newer, better-all-round lossless formats, such as PNG.

Creators leap at animated GIFs, using (and some might argue abusing) the format for more than it was intended for. Quite often on websites with a lot of animation, loading times were (and often still are) very very slow – with some apparently quite small GIFs in the tens of megabytes size range, and tens of seconds loading time. Other more modern formats remain ignored, and whilst browser support for other moving image file formats has improved, not many people have noticed. 'GIF' became synonymous for 'moving picture' as a cultural shift to blogging with moving pictures has become commonplace, especially on platforms like Tumblr, where GIF posts are hugely popular.

The WinFF tool is great for conversion and hooks into ffmpeg. You can find WinFF in the packages for your distribution. In Ubuntu/Debian based distributions, this is simply 'apt install winff'.

However, there's a better way! With browser developers widely supporting video formats such as WebM – the video format for the web, invented by Google – and MP4 out of the box to comply with HTML standards, many image-uploading sites have taken to converting GIFs to silent video.

The silent era

This is looking like it's the way to go, since the equivalent filesize and loading time of a silent video compared to a GIF is significantly lower, often by several orders of magnitude. Videos can also start playing before loading, as they can cache while loading, while GIFs have to load completely before they can play absolutely smoothly.

One item of concern is that some browser developers include the policy to silence auto-playing videos by default, or disallow auto-playing videos with audio. This does not inhibit the moving image use-case of video formats, however, because all we're doing is making them look like plain moving pictures, and retaining the same user experience.

A 95-frame 10fps clip converted from GIF to a silent film format went from 11MB to about 300KB in WebM (about a 30x decrease), and 1.8MB in MP4 (about a 6x decrease) using default options with no noticeable difference in quality. It starts loading in a fraction of the time taken to load the image file, as it does not need to load completely in order to start rendering properly, and there will be no stuttering effect.

» ADJUSTING WEBM QUALITY

A simple way to adjust the quality for WebM is to use Constant Quality options, by adding the following to the option list of either *FFmpeg* before the output filename, or *WinFF*'s FFmpeg options list:

```
-crf 30 -b:v 0
```

Keep the `-b:v 0` option when changing the quality, but you can change the `30` to any number between 0 and 63, where lower numbers mean better quality. Recommended values are from 15 to 35 – 31 allegedly being good for 1080p HD video, according to *FFmpeg* itself. Use the highest value you can that still looks good. Be careful, though, as some values may actually increase the size of the file. In our example, quality 2 kept the size of the file below its original, quality 31 produced the smallest file size with a quality still acceptable to the naked eye, and above 40 began to introduce noticeable video blockiness and artifacts.

As with MP4's Constant Quality settings, it's important to note that this might not be perfect for your particular use case, so look around at the various advanced options for your preferred codec.

A larger list of quality options is available for WebM at: **https://trac.ffmpeg.org/wiki/Encode/VP9**.

You can use either the command line *FFmpeg* or a GUI program such as *WinFF* (don't worry about the name – it's free software which is probably in your distribution's repository) to convert a GIF to a WebM:

```
ffmpeg -i my-gif.gif -an my-video.webm
```

-i specifies the input file. -an specifies that there should be no audio track embedded. The last filename parameter specifies the output file.

After a bit of churning, this will produce a WebM video, suitable for embedding. To embed this into a page, use the HTML <video> tag. Use this template instead of an <img> tag:

```
<video autoplay loop muted>
  <source src="/my/location/to/file.webm"/>
</video>
```

Unfortunately and perhaps not surprisingly, Apple's Safari will not accept this by default on macOS. It requires plug-ins such as those available for VLC and found at **www.videolan.org/vlc/download-macosx.html** (select the 'VLC Web Browser plugin package' option by clicking the down-facing arrow).

Any browser on iOS will also not work with this, since they are all based on the same back-end from Safari – so if you want to add support for these browsers, you'll need to do a couple of further steps and convert your GIF or video to the mysteriously restricted MP4 format. Note that this requires proprietary codecs:

```
$ ffmpeg -i my-gif.gif -an my-video.mp4
```

You'll need to include this via another <source> tag inside your <video> tag:

```
<video autoplay loop muted>
  <source src="/my/location/to/file.webm"/>
  <source src="/my/location/to/file.mp4"/>
</video>
```

You can display a 'play' button by removing the autoplay attribute of the <video> tag; this is recommended to avoid playing potentially flashing images automatically, which could be an epilepsy trigger. You can stop the video at the end like a traditional video by removing the loop attribute.

Remove the muted attribute to allow audio if you did decide to use an audio-enabled video clip – but note that this will not allow autoplaying in certain browser configurations, standards of which are currently in flux at the time of writing.

Embed your code into the HTML of a web page with the <video> tag as described, or otherwise publish to a social media or instant messaging service. You can also use FFmpeg's built-in filters.

MP4 at constant quality 40 increases the size and quantity of its visual blocky effects and artifacts, and at 70 kB is best avoided.

In addition, you can use *FFmpeg*'s extensive range of video effects, which are documented at **https://ffmpeg.org/ffmpeg-filters.html#Video-Filters**. These can be added to both the command line and to the options inside *WinFF*. Things like automated greenscreen effects and merges are available, so this documentation page is worth taking a look at.

If you're wondering if there's anywhere this won't work, 'in the past' is usually the answer. Things are constantly updating, however. According to the website Can I Use..., at the time of writing 97 per cent of all browsers support the <video> HTML tag and the commercial MP4 format, and 89 per cent of browsers support the free WebM format, with the remaining amount mainly able to support these based on the state of their plug-ins. For more info, see: **https://caniuse.com/#feat=video**, **https://caniuse.com/#feat=webm** and **https://caniuse.com/#feat=mpeg4**.

What about other applications? If you have an instant messenger client such as Facebook, Snapchat or WhatsApp for which you record or collate films, you can download the films first to your computer, convert them and then post them to the client, enabling them to load very well on mobile devices. LXF

QUICK TIP

If you're downloading a moving image from a social media or sharing website, ensure it is OK for you to use; the copyright restrictions in each country vary, but it's generally safe to use the Creative Commons search tool when looking for media.

» MP4 QUALITY TWEAKS

A simple way to adjust the quality for MP4 is to use Constant Quality options, by adding the following to the option list of either *FFmpeg* before the output filename, or *WinFF*'s FFmpeg options list:

```
-preset slow -crf 22
```

Presets go from ultrafast to veryslow, with medium being the default. The faster the preset, the lower the quality. crf values are between 0 and 51 – 0 being lossless, 23 the default and 51 being worst. Usually, somewhere between 17 and 28 is fine, 17-18 is usually visually lossless, and every step of six means roughly half the bitrate and file size.

Try a few values and go for the highest that still looks good. Be careful, though, as some values may actually increase the size of the file. In our example, quality 17 kept the size of the file below its original, quality 28 produced the smallest file size with a quality still acceptable to the naked eye, and above 30 began to introduce noticeable video blockiness and artifacts.

It's important to note that this might not be perfect for your particular use case, so look around at the various advanced options for your preferred codec. A larger list of quality options is available for MP4 at **https://trac.ffmpeg.org/wiki/Encode/H.264**.

Credit: https://github.com/xaos-project/XaoS

FRACTALS

Going beyond the Mandelbrot Set

Part two! Missed part one? Grab a back issue on page 66

Mike Bedford continues to review the curious world of fractals by delving into objects that mathematicians first described as "monsters".

OUR EXPERT

Mike Bedford
Bucking the trend, Mike actually believes that maths can be fun, and he continues to be fascinated how such simple rules produce amazing results.

QUICK TIP

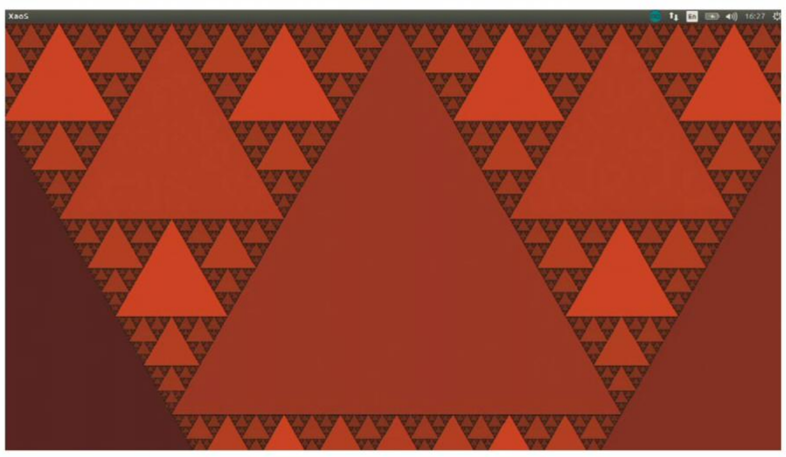
Famously, fractal pioneer Benoit Mandelbrot once asked, "How long is the coast of Britain?" In fact, because of their fractal properties, the apparent length of a coastline depends on the length of the ruler you use to measure it. The smaller the ruler, the larger the result because it's able to follow all the tiny nooks and crannies.

The famous Mandelbrot Set likely springs to mind when talking fractals, but it's just one of a whole group of fractals known as escape-time fractals. As it is there are several categories of fractals that are generated in quite different ways. Escape-time fractals are, perhaps, the most unexpected because, at first sight, it seems barely plausible that applying such a simple formula could produce an image that is visually attractive and literally infinite in the amount of detail it contains. By way of contrast, some of the algorithms used to generate these fractals might seem to be designed specifically to generate a fractal result – but nevertheless we promise to lead you on an astonishing voyage of discovery…

Lost in the Sierpinski Triangle

As fractals are so visual we're going to leave the maths for later and start by providing a hands-on introduction so you can see fractals on-screen from the very start. The fractal we're going to look at is called the Sierpinski Triangle and you can generate it with the *XaoS* software. You can find the Sierpinski Triangle at Fractal > More Formulae > 7 Sierpinski, and investigate it by zooming and panning, using the intuitive interface.

You'll notice the characteristic nature of a fractal – something that shows self-similarity at all levels of magnification – but in a change from the Mandelbrot Set and other escape-time fractals, that self-similarity is exact. Yes, the Sierpinski Triangle contains nothing more than triangles, at ever smaller scales, going on literally forever.



It might not be quite as unexpected as the Mandelbrot Set, but the Sierpinski Triangle is so odd that mathematicians called it a "monster".

The real nature of the Sierpinski Triangle is somewhat hidden in *XaoS*, because of its use of colour to make the image more visually attractive. To see more accurately what this fractal figure really is, and one way it can be generated, head to the online tool at **http://bit.ly/lxf252fracgen**[1]. Initially you'll be shown a fairly small, partially generated Sierpinski Triangle, but we suggest you select a larger size and start with 'Step 0', which is just a single solid triangle. Rather than present you with an apparently complete fractal that you can zoom into, this utility enables you to step through its creation, even though this means that you'd only see the real Sierpinski Triangle if you stepped through an infinite number of iterations.

Now click on the right-arrow icon to move to Step 1 and you'll see that the initial triangle has been split into four smaller triangles and the centre one removed. Move to Step 2, and each of the remaining three solid triangles are similarly divided into four smaller triangles and, in each case, the central one removed. The same process is repeated over and over to produce the Sierpinski Triangle, and it would be a fairly simple coding task to do this yourself.

The Sierpinski Triangle predated Benoit Mandelbrot's coining of the word "fractal". Instead, when this and similar geometrical figures were first discovered in the early 20th century, they were collectively referred to as "monsters". Some of the Sierpinski Triangle's monstrous characteristics are that its perimeter is infinite in length, yet it is contained in a finite area. Moving on, we learn that the area of this geometric figure is zero. Even more strange is that it's a 1.59-dimensional figure. If you thought everything was two- or three-dimensional, or four at a pinch, take a look at the box on page 73.

The Sierpinski Triangle can be described as a geometric fractal, because it's produced by repeating a simple geometric manipulation. However, it might be better to reserve that term to the method of generation because, as you can see from the box on page 71, there are countless ways of producing it.

Nevertheless, if you want to see some more examples of geometric fractals, *XaoS* has a couple more up its sleeve, in the form of the Sierpinski Carpet and the Koch Snowflake.

1) http://codinglab.huostravelblog.com/math/fractal-generator/

Next we're going to delve into Iterated Function Systems, or IFS for short, which is another method of generating fractals. While this approach can be used to generate very regular-looking fractals, it can also produce results that don't look anywhere near as regular and geometric as the Sierpinski Triangle.

Getting iterated

To get a feel for IFS fractals, download *IFStile* from **https://ifstile.com** (you won't find it in the repositories). You could use it to generate very regular fractals of the type we saw in the previous section, but we're going to try something different, so go to File > Open Example > [3D]_Trees and select any of the four trees – we quite liked Tree3. You might think the result looks more like a fern; it certainly looks like something you could find growing in your garden, yet it was produced by repeatedly carrying out a simple transformation. *IFStile* appears to be packed with functionality but unfortunately its Help facility is limited to just providing 'About' information, and the website's link to the user manual didn't work as this article went to press. However, we did find a related PhD thesis by its author, which includes a section on using *IFStile*. It might take some perseverance to really get up to speed with *IFStile*, but if you want to delve into that thesis you can download it from **http://bit.ly/lxf252fractal**.

There's no better way to get a better feel for the IFS approach than writing some software yourself. It's by no means a difficult coding exercise and it's an ideal way to understand how it works. Like virtually all methods of generating fractals, the approach is iterative and, in this case, you start with nothing more than a random point which is defined by its x and y coordinates. The iterative process involves generating and plotting a new point by carrying out a simple geometric transformation that is defined by the following:

$$x_{n+1} = ax_n + by_n + e$$
$$y_{n+1} = cx_n + dy_n + f$$

where x_n and x_{n+1} are the x coordinates of the point in the current and next generation, y_n and y_{n+1} are the y coordinates of the point in the current and next generation, and a, b, c, d, e and f are constants that define a particular fractal. In fact, in all but the very simplest of Iterated Function Systems, there are several sets of those six constants, each of which has a probability in the range 0 to 1 associated with it, the sum of all the probabilities adding up to 1.

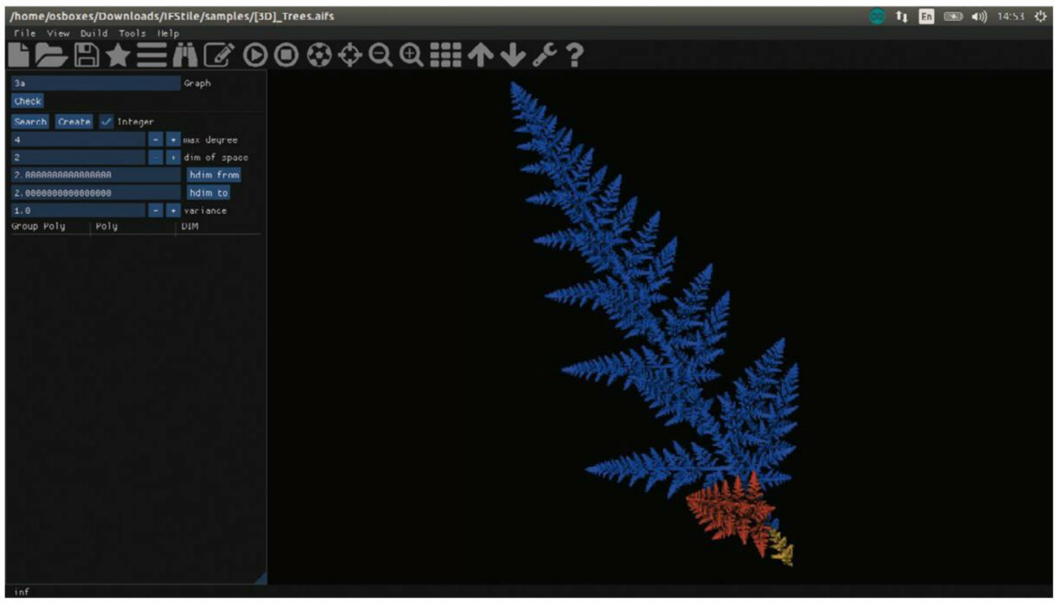
So, for each generation a random number is generated and this is used to determine which of the sets of constants to use. We suggest you start with a known fractal so you know what to expect; for example, the following data generates a fern that looks similar to the one generated by *IFStile*. Each row of the table lists the six constants a, b, c, d, e and f, plus the probability associated with that set of constants, in that order.

0	0	0	0.16	0	0	0.01
0.85	0.04	-0.04	0.85	0	1.6	0.85
0.2	-0.26	0.23	0.22	0	1.6	0.07
-0.15	0.28	0.26	0.24	0	0.44	0.07

The fractal fern is just the tip of the iceberg, so to speak. It transpires that so many of the forms we find in nature have fractal properties and in addition to trees and ferns, we could list coastlines, rivers, clouds, lakes, mountains and even a variety of broccoli called Romanesco, which appears to be a perfect fractal. In reality, of course, anything that exists in nature can only ever be an approximation to a true fractal in the sense that, whereas mathematical fractals go on forever, real-world objects will come to the end of the line eventually, and certainly when molecular dimensions are reached.

The fractal nature of the world around us isn't just an interesting mathematical curiosity but it's also used to good effect in animated movies and video games. Rather than having to store very detailed descriptions of landscapes, realistic-looking islands, mountains, rivers and clouds, they can be generated on the fly using fractal techniques. If you want to try your hand at creating fractal landscapes, options are few. Free software is pretty thin on the ground and most of it is for Windows. However, to get a feel for the results that are achievable, how about trying the demo version of *Fractscape* from **https://starscenesoftware.com/fractscape.html**, which runs under Linux. You'll find that you can generate realistic-looking landscapes, but you can also produce ones that are very otherworldly in their appearance. *Fractscape* is a reasonably fully

CREDIT: https://ifstile.com



The IFS technique is able to produce both natural-looking fractals, like this 'fern', and more regular fractal figures.

QUICK TIP

If you want to delve into yet another aspect of fractal geometry, you could research so-called strange attractors. Head off to http://chaoscope.org for some suitable software.

» A UBIQUITOUS SIERPINSKI TRIANGLE

Here's an interesting experiment you might like to try. You could use a ruler, pencil and paper, and an unfeasibly large measure of patience, although we recommend knocking up a simple bit of software. You could even do it as a spreadsheet. Draw an equilateral triangle, and label the three corners as 1, 2 and 3. Pick a starting point, anywhere inside the triangle, and mark it with a dot. Generate a random number in the range 1 to 3, and find the point halfway between the previous point and the corner identified by the random number, marking it with a dot. Repeat the last step lots of times. Despite the random nature of the process, the end result is far from random. In time, a pattern will emerge, that of the Sierpinski Triangle.

We've now seen two way of generating this image, yet the two methods couldn't be more different. But they're just the tip of the iceberg. Remember the fern that was produced using an Iterated Function system? Well, if you seed the algorithm with different data, it will produce the Sierpinski Triangle. Then we have L-Systems – and we discover that this is yet another way of generating this familiar figure. We're not done yet, however – there are more – but we'll leave you to read up one some of the other ways of generating this most ubiquitous of fractals.

»

QUICK TIP

Of the geometric fractals we've encountered this issue, arguably the Dragon Curve appears the least regular, looking somewhat biological in nature. It might come as a surprise to learn, therefore, that Dragon Curves tessellate. Any number will fit together, leaving no gaps between them.

featured package, even though a licence costs just $20 – vastly cheaper than those packages aimed at games creators and animators. However, it's interesting to note just how little code is required to generate a basic fractal landscape. The utility at **http://demos.playfuljs.com/terrain**, for example, comprises just 130 lines of JavaScript – refresh to see different landscapes, seeded by different random numbers.

Lindenmayer Systems

Aristid Lindenmayer was a Hungarian biologist who studied the growth patterns of yeast, fungi and algae, and developed a mathematical system to define their development. The system that bears his name – but which is often referred to just as the L-system – has more recently been used to model the growth of more complicated biological forms and even geometric-looking fractals.

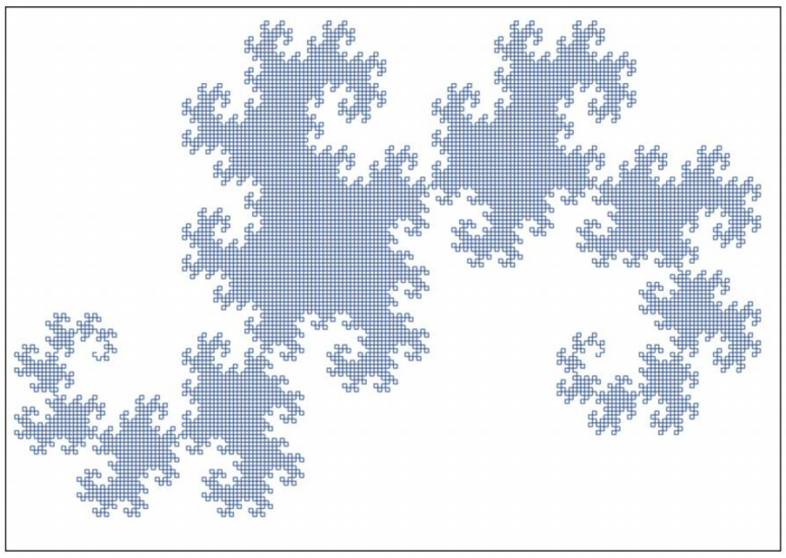
An L-system is a specific type of string-rewriting system in which a particular string of symbols is replaced by another string, generation after generation, each representing a better approximation to a particular fractal. Similar systems can be used as general models of computation, despite the fact that, in this case, the purpose is specifically to generate a fractal figure. A particular L-system is defined by its alphabet – that is, the specific symbols it uses to compose strings, the so-called axiom – that is the string in the first generation, and the set of rules that define how symbols are substituted for other symbols from one generation to the next.

As an example of how this can carry out a general computation, let's think about a system with an alphabet of two symbols, A and B, the axiom of A, and the following two rules: A > AB and B > A. It's a simple task to generate the first few generations by hand and you'll discover that they are A, AB, ABA, ABAAB, ABAABABA, ABAABABAABAAB, and ABAABABAABAABABAABABA. Although this particular system was developed by Lindenmayer for modelling algae growth, it's evident that the lengths of the strings – 1, 2, 3, 5, 8, 13, 21... – are actually Fibonacci Numbers: that is, the sequence of numbers in which each number is the sum of the previous two numbers.

For the purpose of generating fractal images, we use an alphabet in which each symbol represents a drawing function. We'll illustrate this to draw a geometrical-

The demo version of Fractscape illustrates how small amounts of data can be used to generate realistic-looking landscapes.

CREDIT: http://starscenesoftware.com/fractscape.html



Using a Lindenmayer System is just one of several ways of generating the Dragon Curve.

looking fractal that we've not seen so far. The alphabet comprises F and G, both of which are instructions to draw a line of a fixed length in the direction of the current heading; + which is an instruction to turn 90 degrees to the right; and – which is an instruction to turn 90 degrees to the left. The axiom is F, and the rules are F > F - G and G > F+G.

Like so many fractal coding exercises, it couldn't be much simpler to try this out in your own code. One bit of advice we should give is that the length of the line generated by an F or G command should halve with each generation, so that the overall figure remains approximately the same size. We also suggest that you write the code so that you can move from one generation to the next via a keystroke, to appreciate how the fractal develops. The result is called a Dragon Curve and, if you'd rather use someone else's code, there's a good online utility for a wide range of fractals at the same link we saw earlier at: **http://bit.ly/lxf252fracgen**[1].

By default this shows the Sierpinski Triangle, but the Dragon Curve is just one of its many fractals, and it allows you to track it from generation to generation, starting with just a straight line. Having seen the Dragon Curve build up, it's clear that the process of moving from one generation to the next involves replacing all straight-line segments with two shorter line segments and a right-angled bend, and this suggests another way of generating it, if you fancy knocking up a bit of code.

It might not involve coding, or using any software at all for that matter, but you might be interested to learn of another, very different way, of generating the Dragon Curve. Take a sheet of A4 paper and observe the longest side, edge on. What you'll see, of course, is a straight line, which just happens to be the first generation of the Dragon Curve. Now fold the paper in half, with the fold across the shortest edge, and then open it out so the crease is a right-angle. Looking at this edge-on again, you're now seeing the second generation. Fold it back so it's flat and then fold it again, parallel to the first fold. Again open it out so that the creases are right-angles and you'll now observe the third generation of the Dragon Curve.

As you make more and more folds, it becomes increasingly difficult to open the sheet up so that all the folds are right-angles, but you might like to go as far as you can so that you get a more representative Dragon Curve. Although this has been an exercise in origami rather than programming, the process can easily be expressed algorithmically, so it offers yet another way of generating this fractal in software.

The Dragon Curve can be represented as a string of left and right bends, which we can show as R and L, with an implied straight-line section between each bend. Using this way of expressing it, the first generation is the null string, which represents a straight-line section with no bends. Now to generate the next generation, start with the string for the previous generation, add an R, and then append the string for the previous generation reversed and with all the Ls converted into Rs, and all the Rs changed to Ls. This, therefore, will give R as the second generation. Successive generations are represented by RRL, RRLRRLL, RRLRRLLRRRLLRLL and so on.

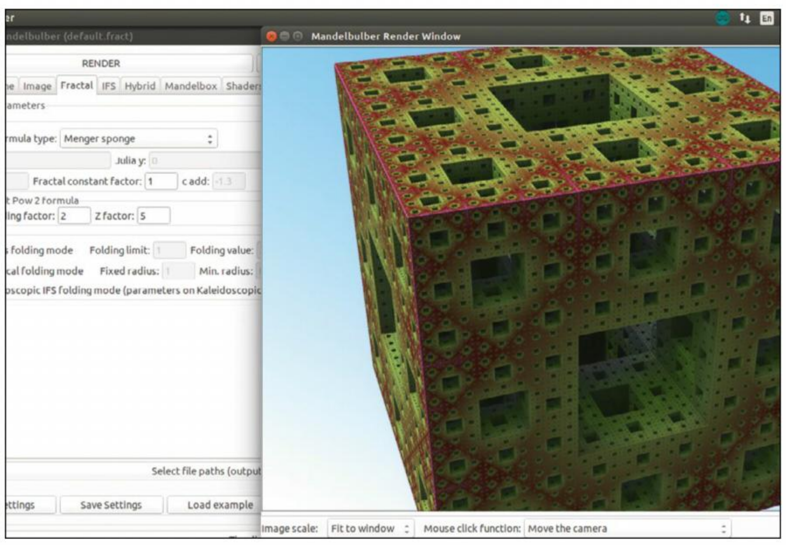
Finally, for those who intend to write some code, whatever method you use for generating the Dragon Curve you might like to consider rounding off the right-angled bends. The curve never crosses itself but it does touch and this can give the impression of it crossing. If all right-angles are rounded off, however, the true path is evident – plus you might feel that this representation is more aesthetically pleasing.

Into the third dimension

In the light of our coverage last issue of the Mandelbulb, – the 3D equivalent of the well-known 2D escape-time fractal the Mandelbrot Set – it seemed appropriate to round up this article by extending the principles of the more regular fractals into the third dimension as well. In fact, since you might already have it installed last issue, the obvious choice of software is *Mandelbulber* – and as one of the fractals we've seen most of in this article, why not take a look at the 3D equivalent of something rather similar to the Sierpinski Triangle?

So, start up *Mandelbulber*, select 'Menger sponge' as the Fractal formula type, and click the Render button. The fractal of that name appears in the Render window. Quite evidently, it's been created by repeatedly splitting a cube into smaller cubes and, after each such sub-division, removing one of the cubes – and in that respect it's a close relative of the Sierpinski Triangle. This is by no means a unique example. 3D fractals abound; indeed the fractal landscapes we looked at briefly are another example. What's more, some of the more advanced landscape generation software is able to populate those fractal landscapes with forests made up from fractal trees.

In fact, why stop at three dimensions? We might live in a three-dimensional world but that doesn't stop mathematicians postulating about four-dimensional figures, and that includes fractals. The only snag is that they're hard to visualise, so we have to make do with three-dimensional projections – and even those have to be rendered in two dimensions for display on screen. However, if you fancy a more challenging programming challenge, why not respond to the call of the fourth dimension? (And we don't mean time in this case!) LXF



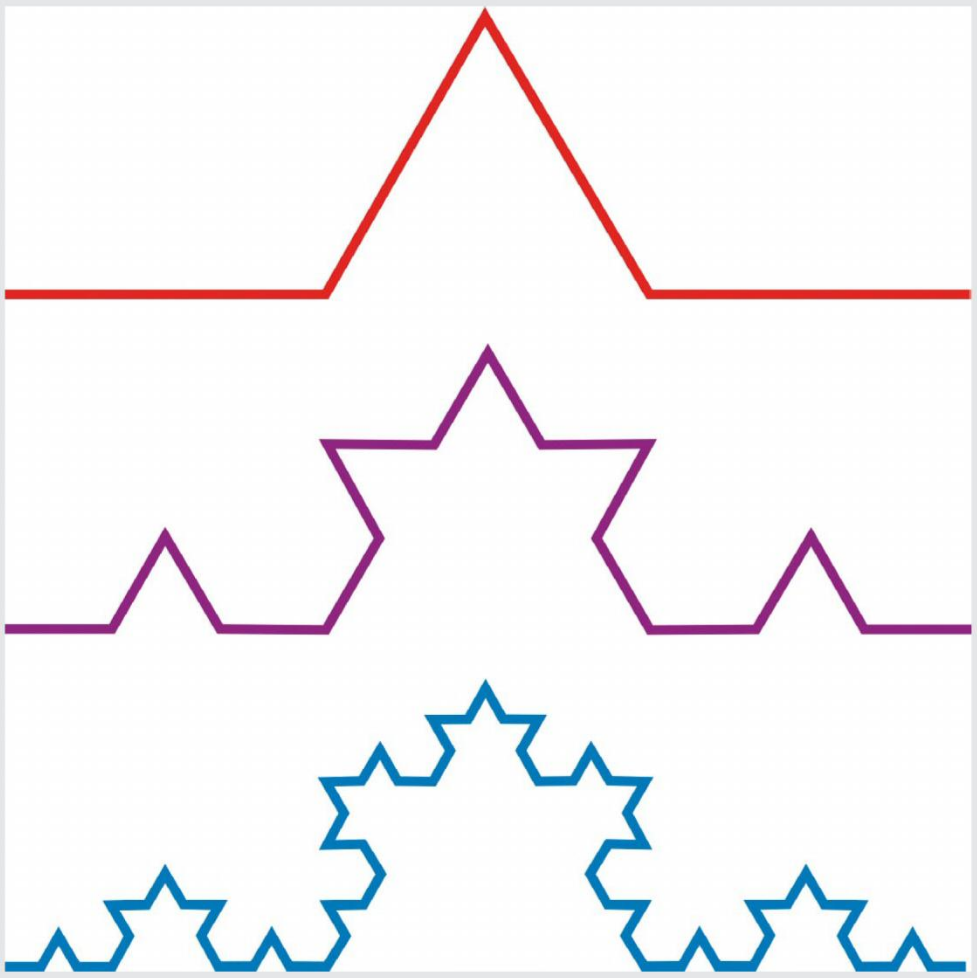
CREDIT: https://github.com/buddhi1980/mandelbulber2

The principles of generating geometric fractals can extend into the third dimension and beyond.

» FRACTAL DIMENSIONS

We're accustomed to thinking of lines and curves as one-dimensional, planes and surfaces as two-dimensional, and solid objects as three-dimensional. The number of dimensions is a measure of how many figures we need to define a point in the object. So, with a line or curve we need just one – the distance along it; with a plane or surface, we need two; and with a solid object we need three. With fractal figures all this changes, and we find that the number of dimensions is fractional.

Take the Koch Curve (which isn't the same as the Koch Snowflake), for which the first few iterations are shown below. It has a behaviour intermediate between a curve and a surface. If we look at any point on the curve – the true Koch Curve, that is, not an approximation shown here – it looks nothing like a line. Instead, it shoots off in every direction, and in that sense it more resembles a surface. In fact, its number of dimensions is intermediate between 1 and 2. Specifically, and bizarrely, it's a 1.269-dimensional figure. If you find that strange, we'll leave it as an exercise for you to read up on.



The Koch Curve can't be described as curve or a surface, which is why it has a bizarre fractal dimension of 1.269.

Credit: https://github.com/sa2c/sombrero

SOMBRERO

Comprehensively benchmark your PC

Ed Bennett shows how you can apply the same techniques used to benchmark the world's fastest supercomputers to one-up your friends…

OUR EXPERT

Ed Bennett
When not speed-testing supercomputers, Ed helps and trains academic researchers.

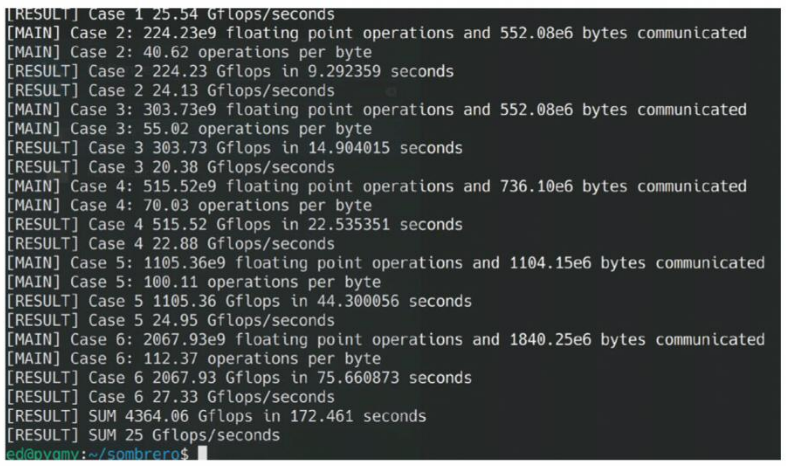
QUICK TIP

Running across multiple machines works best if you have the same username on all of them. If you don't, you'll need to put the software in a shared directory, and create a ~/.ssh/config file to let Open MPI know how to connect to each machine.

You've probably seen benchmark statistics reported about all manner of popular devices, but what exactly is a benchmark, and how can you run one on your own systems?

A benchmark originally referred to a testing process for rifles, where the gun was mounted to a bench, rather than being held by a human marksperson: the bench was acting as the mark, hence benchmark. With several test firings, the spread of shots could be measured – the smaller the spread, the better the gun. Nowadays however it refers to any process that allows different methods, devices or processes to be compared with each other directly, based upon a standard reference.

In computing, this is done by taking a set of reference tasks which reliably take exactly the same amount of effort each time, and measuring how a system performs when completing them. These tasks might be designed to stress one specific component – for example, hard disk write speed; the machine, to see what the absolute limits of the system are (synthetic benchmarks); or they might be representative of the kinds of work that a system will be used for (application benchmarks). A gaming benchmark might run a set portion of game over a fixed route and measure the frame rate, while a 3D rendering benchmark may choose a complex 3D scene and measure the time the computer takes to render it. Benchmarks are frequently distributed as benchmark suites; since it's rare for a computer to only be used for a single task, benchmark suites measure all-round performance by including contributions from a variety of typical tasks.



A run of Sombrero on a two-core laptop with an Intel Core i7-7660U takes around three minutes.

In this article, we'll look at the *Sombrero* benchmark suite. *Sombrero* is a suite of application benchmarks for a piece of high-performance research software used in theoretical particle physics computations. The benchmarks within it stress to varying degrees the raw speed of the processor, the speed at which data can get into the processor from the system memory, and the speed at which data can be communicated between the different CPU cores and computers that are cooperatively working on a problem.

Despite some effort to promote alternatives, in the world of high-performance computing the command-line reigns supreme. Because of this, the instructions that follow all take place at a terminal. To start, you'll need to install the dependencies: *Git*, *make*, *GCC*, *bc*, *gnuplot*, *OpenSSH* and *Open MPI*. On Ubuntu, these can be installed with:

```
$ sudo apt install git make gcc bc gnuplot \
 openssh-server openmpi-bin openmpi-common
```

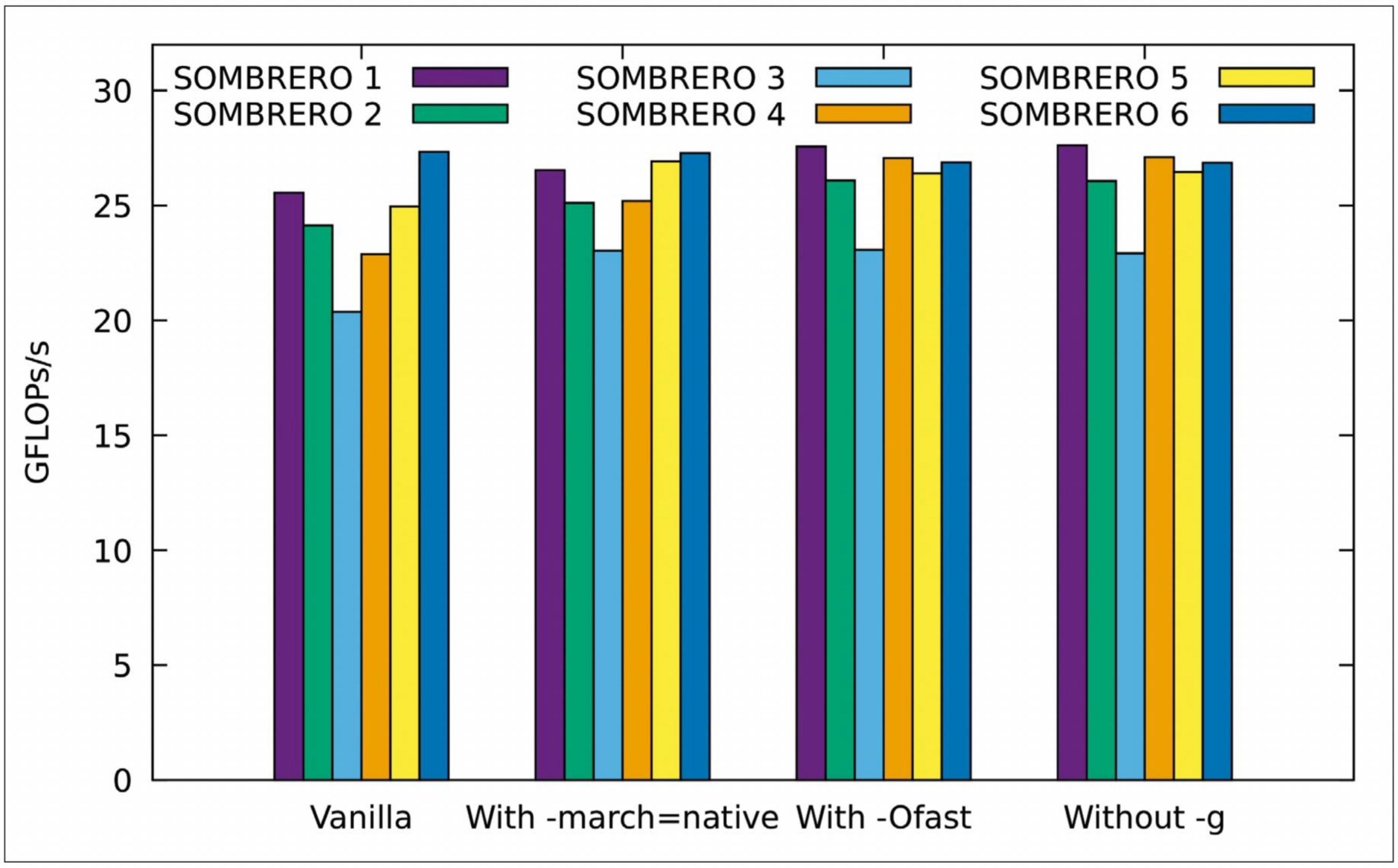
Since you've installed *Git*, you'll be able to update to the latest version of *Sombrero* from the repository at **https://github.com/sa2c/sombrero**:

```
$ cd ~/sombrero
$ git pull
```

The default settings are enough to get you up and running, so type `make` and *Sombrero* will build itself. It includes six different benchmarks, each from a different formulation that theoretical physicists are studying to see whether or not it could describe reality, so while *Sombrero* builds you'll see the same files compiled six different times, once for each benchmark. Also included is a shell script that runs each benchmark in turn, and gives a summary of the results. Since *Sombrero* is a highly parallel benchmark suite designed to run on some of the largest computers in the world, you need to tell it that you want to run it on a single CPU core, and to use the smallest problem size it knows about. To do this, use:

```
$ ./sombrero.sh -n 1 -s small
```

You'll immediately see some output from the first benchmark about the setup – since this test does not communicate between processes at all, the 'operations per byte' is infinity. Within a couple of minutes you'll see the results of each benchmark start to appear. The headline figure is the second `[RESULT]` line within each block—the one ending `Gflops/seconds`. 'Gflops' here is



On this laptop, -march=native gives a noticeable bump to almost all benchmarks, -Ofast gives a small speedup to some, and removing -g has little effect.

short for gigaflops, or 'billion floating point operations' (where a 'floating point operation' is any operation such as addition or multiplication done on a number with a decimal point). Since most supercomputers are used to crunch numbers for science and engineering applications, this gives a pretty good comparator between different machines.

Get parallel

So far this is only using a single core of your CPU. Since you'll be hard-pressed to find a computer with only one core in 2019 (*Jonni's Eee PC–Ed*) a good next step is to run using every core in your machine, to get the highest number you can. To find out how many cores your machine has, you can check the specs, or you can look in **/proc/cpuinfo**. (If your CPU has hyper-threading, you'll need to divide by two, since *Open MPI* doesn't believe that those are real cores.) Put the number you get in place of the `1` in the above command to run on all cores. For example, for a six-core workstation, run:

```
$ ./sombrero.sh -n 6 -s small
```

It's worth noting that because of the particular problem *Sombrero* is based on, it can only test a number of cores that is divisible only by 2 and 3. For example, 6 (2 × 3), 24 (2 × 2 × 2 × 3), or 32 (2 × 2 × 2 × 2 × 2) will work, but 10 (2 × 5) or 14 (2 × 7) won't work, and will give you an error.

Now you've successfully brought your entire machine to a halt, how about doing the same with your entire network? If you have more than one Linux machine joined by a network, you can try running *Sombrero* across every machine you have access to, forming a makeshift Beowulf cluster. First off, follow the instructions up to now for every computer that you want to participate. This will check that *Sombrero* works on each machine individually, and also set up the executable so that *Open MPI* can find it on each machine. Next, you need to configure SSH so you can log in from one machine to another without a password.

To activate the SSH server, run

```
$ sudo systemctl start ssh
```

On Red Hat-derived distributions, this will be `sshd` instead of `ssh`. Then on each machine in turn, generate an SSH key by running

```
$ ssh-keygen
```

and pressing Return three times. Once this is done, you can copy the key to your other computers with

```
$ ssh-copy-id destination_machine
```

where `destination_machine` is the hostname or IP address of the machine to copy to. When prompted, enter your password for the destination machine to let the copy happen. If you have two computers you'll need

QUICK TIP

If you're running on a Raspberry Pi, even the 'small' test is too big to fit into the system's 1GB of RAM. You can specify a custom size: try -l 16x12x12x12.

» WHAT IS A SUPERCOMPUTER?

A modern supercomputer isn't too unlike a home or office network. A bunch of Linux servers (or nodes), often using Intel processors and mounted in racks, can all operate independently, but sit on a common network. In the fastest supercomputers, each server has one or more GPUs installed to perform the heavy grunt of the computation. Red Hat Linux is the most common distribution used, with Debian-derived distributions being more of a rarity.

While a typical Ethernet connection may be used for managing the nodes and logging into them, they are also connected together by an ultra high-speed interconnect such as Infiniband or Omni-Path. The nodes need to work cooperatively on large problems, and don't want to sit idle waiting for data to arrive, so planning the network layout is a crucial part of designing a supercomputer. While most machines will use a relatively familiar hierarchy of switches, some machines targeting specific applications will design the network around that, giving exotic shapes like dragonflies and six-dimensional toruses!

Intel CPUs will be of the high-end server varieties, since this gives a much higher number of cores per node: current-generation chips give up to 24 cores on a single chip, compared to eight on the highest-end Core i9. GPUs will be computationally-focused NVIDIA Tesla cards – often with no display output at all.

QUICK TIP
If you don't have more than one machine to play with, you can make a comparison plot of any parameters you can tune. One particularly useful plot is to compare different compiler options, to see which is fastest.

to do this once on each, if you have three it will be twice on each, and so on. This is because MPI on each machine needs to be able to talk to every other machine, and you won't be able to type in the password once the program is running.

Now you'll need to let *Open MPI* know what machines you want it to use. To do this, create a text file called **hostfile.txt**, and in it write down the hostname of each computer you want to use. Repeat the name as many times as you want processes to run on that machine; for example, if you want your dual-core laptop called `pygmy` to join in a computation with a six-node workstation called `pyxis`, **hostfile.txt** will look like:

```
pyxis
pyxis
pyxis
pyxis
pyxis
pyxis
pygmy
pygmy
```

With the setup all done, you can test your new cluster with the command

```
$ ./sombrero.sh -H hostfile.txt -s small
```

In an ideal world, this will be faster than the score for one computer. That's not guaranteed, though – that depends on how fast your network is, and how fast the other computers are. Since the individual steps of the benchmark have to happen in lock-step across the entire cluster, the progress will be dictated by the slowest machine. Because of this, removing a particularly slow computer from the mix may enable things to speed up.

This is a problem that occurs frequently in 'heterogenous computing'. For many problems, load-balancing techniques can be used to adjust the amount of work each machine does based on its capability; however, most benchmarks don't bother with this, both because the underlying problems don't lend themselves to it, and because the systems that will be tested generally have very uniform components.

Pursue the need for speed

So far, you've been running *Sombrero* with the default compiler settings, which are `-std=c99 -g -O3`. Now, those are fine settings to use for testing, but won't get the highest possible performance numbers for your system. To get this, you need to give the compiler more directions on what you want it to do to get the best performance. You do this by editing the **Make/MkFlags** file, which defines variables to be read by the compiler. Opening this file in your favourite text editor, you'll see:

```
#Compiler
CC = mpicc
CFLAGS = -std=c99 -g -O3
LDFLAGS =
```

Since we installed *GCC* earlier, you might expect to see `CC = gcc`; instead, `mpicc` is used. `mpicc` is a wrapper around *GCC* that *Open MPI* provides, and makes sure that all the correct communications libraries are linked.

To start with, the `-g` flag enables debugger symbols. This is useful when you're debugging problems with software, but can in some cases slow down performance slightly. Remove this flag now, and run `make` again. Re-running *Sombrero* now, either on your local machine or across a cluster, should reveal a slight speed increase compared to the previous run. The next step is to tell the compiler what kind of processor you're going to run on.

By default, *GCC* only allows itself to use processor features that are on a generic x86-64 CPU – this limits it to features from processors released in 2003. CPUs have come a long way since then, so enable *GCC* to use these new features by adding the flag `-march=native`. If you want to run on a different machine than the one on which you're compiling, you can pass other options than `native` to the architecture parameter; the *GCC* manual has the full story. Re-compile with `make` and re-test now, and you should see a more significant increase in performance.

GCC has hundreds of knobs you can tune to try and extract the best possible performance from your machine; you can browse the full list at **https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html**. This might seem like cheating, but it's an important part of both benchmarking and running software on large machines. The compiler is designed for a wide range of software and hardware, so sometimes needs help to know how best to make your software run. On a typical desktop, the performance difference is small enough that compiling every piece of software from scratch and testing every option will spend more time than it saves. But when the software you're working with is running on hundreds or thousands of identical machines for weeks or months at a time, you want to make sure that you're not wasting even five per cent of the available power.

In more extreme cases, a hardware manufacturer will rewrite all or part of a benchmark to be faster on their hardware. Again, while this seems unfair, in some cases

» TESTING AT SCALE

Since supercomputers rely heavily on their high-speed network to let computations be distributed across many nodes, it's important to test how well this works. There are two ways this can be done: weak scaling tests, where the problem size grows with the number of nodes used, and strong scaling tests, where the same problem is divided into smaller and smaller chunks. *Sombrero* supports both kinds of tests, defaulting to strong scaling, since that is the harder problem.

If you have enough computers available, you can run a strong scaling test. First, set up a set of node lists, called **nodelist1.txt**, **nodelist2.txt** and so on, using multiples of 2 and 3.

```
$ for NODES in 1 2 3 4 6 8; do
 ./sombrero.sh -H hostfile${NODES}.txt -s small | awk \
  'BEGIN {printf("'${NODES}'");}
   /^\[RESULT\] Case.*Gflops.seconds/ {printf("\t" $4);}
   END {printf("\n");}' >> strong.dat
 done
```

Then, to plot the results:

```
gnuplot> set style data linespoints
gnuplot> plot for [i=1:6] 'strong.dat' \
using 1:(column(i+1)) title 'SOMBRERO '.i
```

The ideal case is a straight diagonal line; in reality, the more nodes are used, the further from straight the line gets, as more and more communication has to happen.

it's necessary to let the benchmark run at all. For example, *Sombrero* currently can't run on GPUs; if a manufacturer wanted to show off the power of their GPU-based supercomputers, they would need to adapt it so that it could run there. This is allowed, provided that the problem being solved doesn't change; that would be a different benchmark entirely.

Plotting the results

What good is a set of benchmark data without a set of colourful graphs to show off? While you could copy and paste numbers into *LibreOffice Calc* by hand, it makes a lot more sense to use a tool specifically designed for text-based data, so let's get started with *gnuplot*.

For this, the first thing you'll need to do is transform the results logs into a more tabular format. You can do this using *awk*. To start with the previous example:

```
$ ./sombrero.sh -H hostfile.txt -s small | awk \
'BEGIN {printf("pyxis and pygmy");}
/^\[RESULT\] Case.*Gflops.seconds/ {printf("\t" $4);}
END {printf("\n");}' >> results.dat
```

This will append one line to the end of **results.dat**, with seven columns separated by tabs. The first will contain the label `pyxis and pygmy`, with the remaining six containing the results of the six benchmarks. You can repeat the process for various configurations by editing **hostfile.txt** – perhaps pyxis by itself and pygmy by itself for comparison. Change the value of the `label` variable so that each line gets a unique label.

Now that your results file contains a table of results, you're ready to generate a plot. Start *gnuplot* with:

```
$ gnuplot
```

This puts you into the *gnuplot* interpreter, with a prompt that looks like `gnuplot>`. Tell *gnuplot* that you're going to plot a grouped bar chart, with filled bars and a black border:

```
gnuplot> set style data histogram
gnuplot> set style histogram cluster gap 1
gnuplot> set style fill solid border rgb "black"
```

Next, ask it to automatically scale the horizontal axis, but make sure that the vertical one starts at zero. Bar charts starting away from zero are generally used as the tool of those who want to mislead you about the data they're presenting!

```
gnuplot> set auto x
gnuplot> set yrange [0:*]
```

Then make sure that *gnuplot* knows that the data file uses tabs as its separator, since by default it will use any white space to separate values:

```
gnuplot> set datafile separator "\t"
```

Graphs aren't particularly useful without axis labels to tell you what the information means. Add one with:

```
gnuplot> set ylabel "GFLOPs/s"
```

Finally, you're ready to plot the data. This command makes use of a loop, so that you don't have to type the same commands over again for each column:

```
gnuplot> plot for [i=1:6] 'results.dat' \
using (column(i+1)):xtic(1) \
title 'SOMBRERO '.i
```

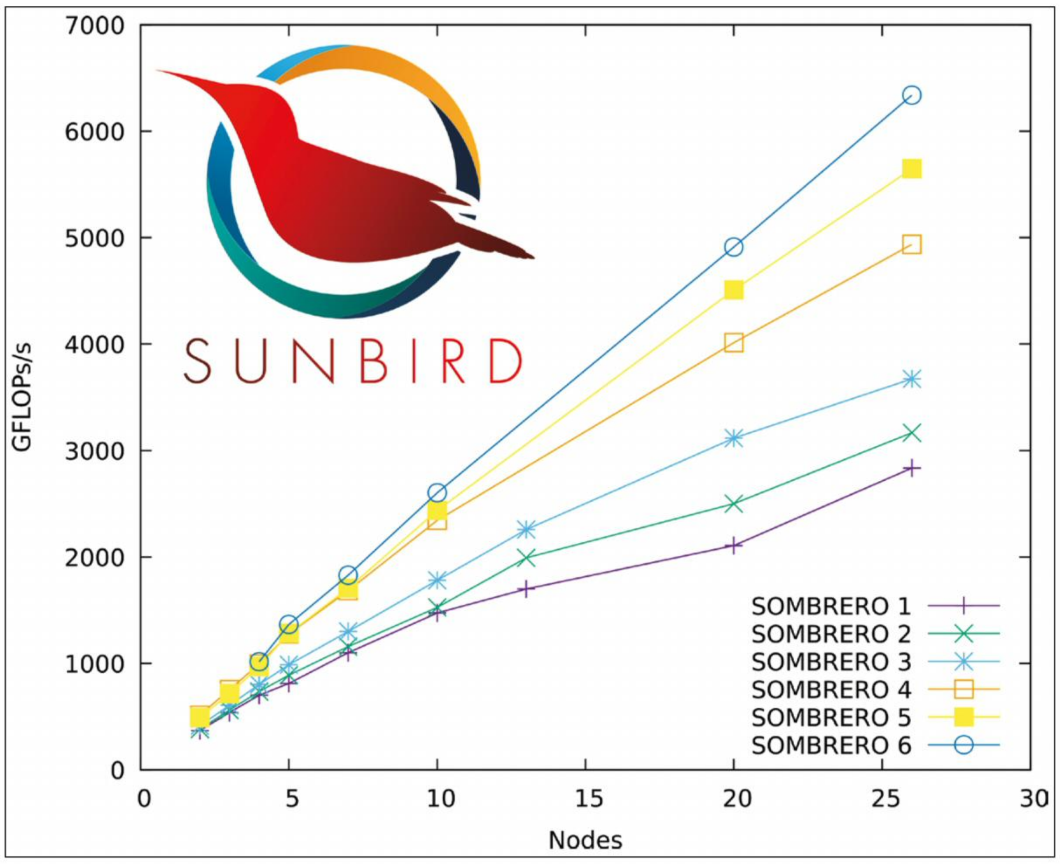
And with that, you're done! A plot appears in a new window, showing a labelled comparison of your different runs as a grouped bar chart. If you want to, you can export it as a PDF or image file, to email to your friends to show off, or print out and hang on your fridge.

In this article, we've explored how you can benchmark your system and even your entire network by using the *Sombrero* benchmark. So next time someone claims to have a faster setup than you, or you're trying to justify buying a new one because your machine has slowed down in its old age, you can back up your arguments with cold, hard data. LXF

» TOP TRUMPS FOR SUPERCOMPUTERS

Once upon a time, the *Linpack* benchmark – used to calculate the Top500 (**www.top500.org**) – was the only benchmark in town for ranking supercomputers. However, since that benchmark is designed to do as many floating point operations (FLOPs) and as little else as possible, it doesn't always paint a good picture of how well a machine will perform when it tries to do something more useful. Hardware manufacturers want to sell more hardware, and they sell hardware by having the biggest numbers, so they design machines to do better at *Linpack* than anyone else. Because all hardware design involves compromise, this can have the perverse effect of harming the machine's performance in properly interesting problems, just to get a better TOP500 score.

Because of this, a whole zoo of benchmarks and rankings has started to spring up. For example, Green500 measures supercomputers' energy efficiency when running *Linpack*: crucial, when as a rule of thumb the electricity costs for running a supercomputer for its lifetime are about the same as the cost of buying the machine. Graph 500 (**http://graph500.org**) focuses on data-intensive applications rather than the computationally intensive *Linpack*. *Sombrero* focuses on a mix of raw computation, speed of reading from RAM, and time it takes to send data across the high-speed network between nodes – with the balance between these shifting between the different benchmarks. For each benchmark, a different supercomputer will reign champion.



The Sunbird supercomputer at Swansea University shows good strong scaling when running Sombrero. The high-numbered tests give near-ideal scaling.

HotPicks

Hyper » DiffPDF » KWin-lowlatency » LanguageTool » Kepka » Password Safe » Devede NG » SuperTuxKart » OpenPatrician » Stacer » WallGen

Alexander Tolstoy
escapes the heat and finds himself discovering even more great open source apps in a cool shady place.

TERMINAL EMULATOR

Hyper

Version: 3.0.2 **Web:** https://github.com/zeit/hyper

There's no lack of terminal emulators for Linux, but fortunately this is a really smart one. *Hyper* is coded entirely with HTML, CSS and JavaScript, powered by Electron. At the same time it's reasonably compact and very robust. Of course this is beneficial for any application, but in the case of *Hyper* we have some remarkable extra features that make it stand out.

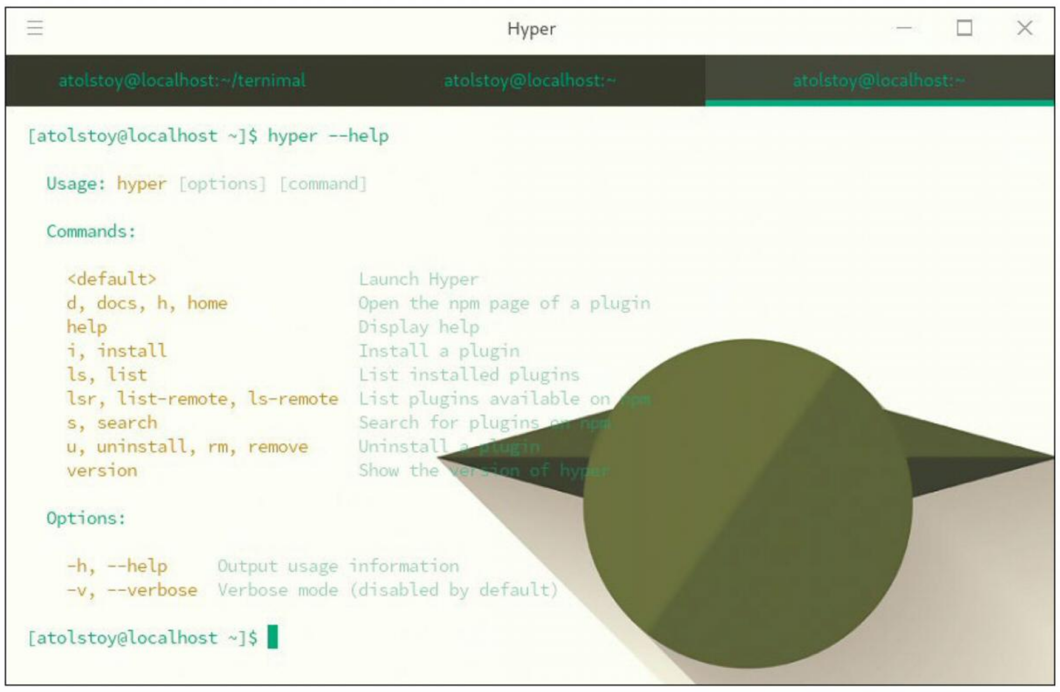
The first is performance. Don't look at the high memory consumption – every Electron app seems to suffer from this. In real-world cases, *Hyper* sports shorter input latency and faster text output. The easiest way to prove it is to run something like `$ time cat big_file.txt` in both *Hyper* and your current terminal, and compare in real time. Older versions of *Hyper* used to be buggy and fell short when compared to other performance-tuned terminals such as *Alacritty* (**LXF241**), but the 3.x series release is much more stable and usable, thankfully.

In addition, *Hyper* has a built-in plug-in manager, which gives you access to more than 20 extra features. Installing a plug-in involved running a simple command like this:

```
$ hyper i hyper-search
```

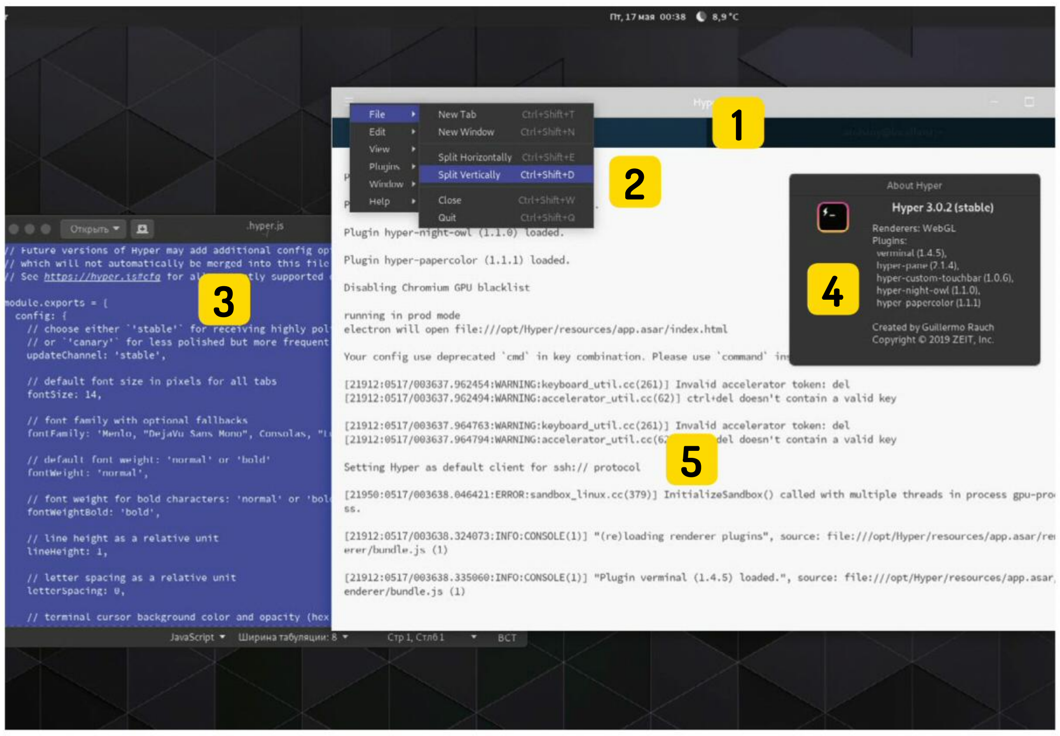
To see what other plug-ins are available, use the short `$ hyper s` command. It's possible to customise *Hyper* for any workflow by using plug-ins combined with the built-in features, such as splitting the main panel into a multiplexed view. Optionally you can edit the **.hyper.js** file, which stores a template of all current parameters that affect behaviour and look and feel.

But the main reason for staying with *Hyper* is again its fluidness and responsiveness. Not that other terminals are slow, but *Hyper* feels a little smoother – maybe thanks to its WebGL support, which it inherits from Electron. Grab the DEB/RPM/AppImage file from the project's download page and give it a try if you're not already convinced – you've got nothing to lose and a whole lot to gain.



Make Hyper a bit more personal by installing a custom theme from the list of plug-ins.

Exploring the Hyper interface



1 Tabs support
Press Ctrl+Shift+T to open a new tab. There's also a plug-in that will make it open to your current directory.

2 Go multiplexing
Break one Hyper terminal into many.

3 Edit preferences
The configuration file is the place to change fonts, adjust letter spacing and set custom colours.

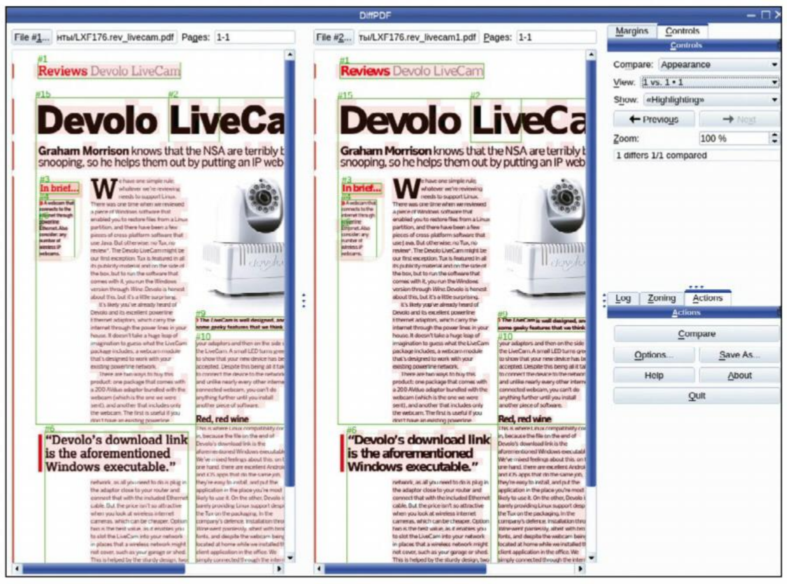
4 Enjoy fast rendering
Hyper automatically selects the Canvas or WebGL renderer to deliver a very smooth visual performance.

5 Use the '$ hyper' command
Append '--help' to learn more, or use '-v' to review Hyper's JS internals.

PDF COMPARE TOOL

DiffPDF

Version: 2.1.3.1 **Web:** https://gitlab.com/pianoslum/diffpdf



Even the tiniest distinction doesn't evade DiffPDF.

Despite the fact that PDF came from the world of commercial publishing and the imaging business, the amount of open source for handling PDFs is impressive these days, and no one could say that Linux is behind the curve in this regard. We've looked at many PDF viewers, editors, splitters and whatnot before, but we haven't seen a PDF compare tool yet.

Why not finally fill that gap with *DiffPDF*? This is an exceptionally useful thing that helps detect and locate differences between nearly identical PDF files. The interface is welcoming and easy to use; the main window is split into two parts, with extra options and controls residing along the right side. To start, load a PDF file in the left and right panels and hit the Compare button. Any parts of the files that do not match are highlighted and you can see them with no effort.

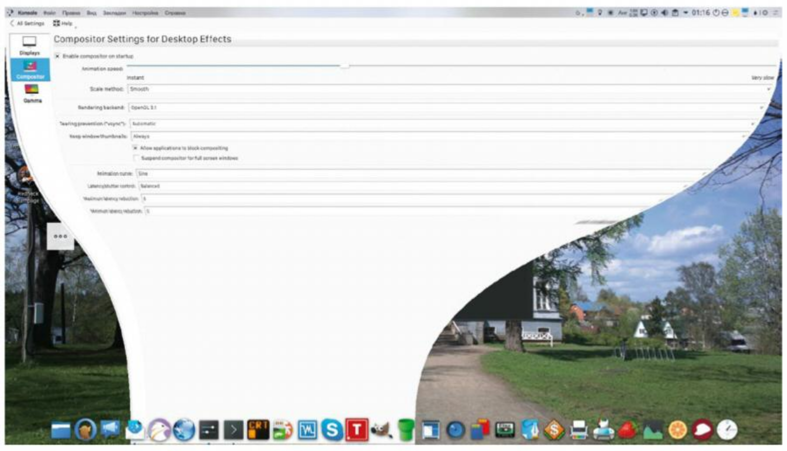
However, the power of *DiffPDF* is not just that ability to compare, but also in enabling you to choose what exactly you want to compare inside files. By default, *DiffPDF* compares appearance, which means that if we open a PDF in an application like *Inkscape* and simply re-save it with automatic font substitution, *DiffPDF* will treat the copy as a totally different file. But once we select Characters or Words under the Control section in the top-right part of the *DiffPDF* window, those pink highlights will be gone. Selecting the comparison method allows for more granular and precise results, with more objective highlighting.

DiffPDF also has a secret card up its sleeve! Navigate to the Zoning tab and enable the same-named option to make *DiffPDF* respect the objects' placement inside PDF files. Zoning can be used to detect things that no eagle-eyed reader could ever find: variation in how text blocks, frames, images and other objects are sliced inside the PDF layout. If you are not sure which of your same-looking PDF files are newer or better, try *DiffPDF*.

WINDOW MANAGER

KWin-lowlatency

Version: 5.15.5 **Web:** https://github.com/tildearrow/kwin-lowlatency



A still picture cannot show how silky-smooth this effect now is.

This *Hotpick* will probably be a reason to rejoice for all KDE Plasma users who are not (quite) happy with the performance of KWin, Plasma's window manager. In fact, KWin in its current state is already great and stable, but given the fact that it incorporates sophisticated code for GPU acceleration and some mind-blowing visual effects, your mileage may vary.

KWin-lowlatency is a fork of the upstream KWin project with extra patches designed to improve responsiveness, remove unwanted latencies and eliminate Vsync-related issues such as tearing. This pick is also very hot: at the time of writing it supports only the latest two versions of Plasma, 5.14. and 5.15, and therefore is targeted at those who run bleeding-edge Plasma versions (think Arch, Tumbleweed and so on).

If you have a Linux box with a fresh Plasma version, give *KWin-lowlatency* a try and see how it goes. The code takes a noticeable fraction of time to get compiled using the convenient *cmake*-based approach (have a cup of tea in the meanwhile). Once you have the build, run `sudo make install` and do a logout/login sequence. *KWin-lowlatency* replaces the stock KWin version and therefore you'll be using the new version with no extra effort.

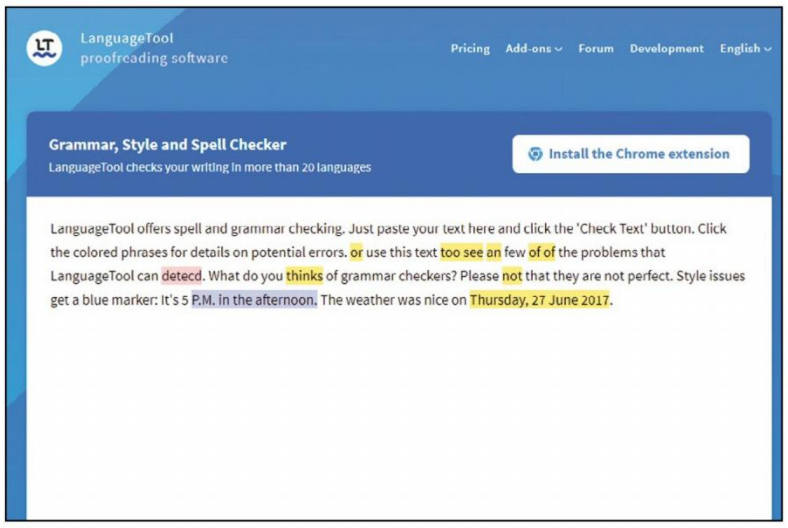
Users of Plasma 5.15 and onward will notice the updated section under System Settings > Display and Monitor > Compositor, with some sweet extra tweaks that don't exist for regular KWin versions. Play some OpenGL games or launch an HD video to see if it runs smoother than before. *KWin-lowlatency* uses glXWaitVideoSync instead of DRM VBlank, which implies better graphics performance regardless of your video driver.

The patches that make it possible are considered experimental and are therefore not included in the upstream project, but they already do a great job for many configurations. If something does go wrong, revert with `sudo apt-get install --reinstall kwin-x11`.

SPELLING AND GRAMMAR CHECKER

LanguageTool

Version: 4.5.1
Web: https://www.languagetool.org



We all make errors, so why not try to eliminate them using the powers of LanguageTool?

LanguageTool is an unmatched companion for anyone who writes a lot and needs to quickly fix spelling or grammar errors. Note that this applies to both native English speakers and others, since it supports more than 25 languages. The biggest benefit to using *LanguageTool* is that it provides both a regular spell-check and a set of rules for fixing grammar errors. Commas, articles, tenses, prepositions and a lot of other linguistic hard bits immediately get highlighted with *LanguageTool*'s famous blue curvy line once a user commits a mistake.

Although there is a handy proofreader tool right at the project's main web page, *LanguageTool* has many other usage options. The most evident and easy way is to install the extension for *LibreOffice* or *OpenOffice*. There is also a standalone Java-based application and a collection of browser extensions for *Chrome/Chromium* and *Firefox*.

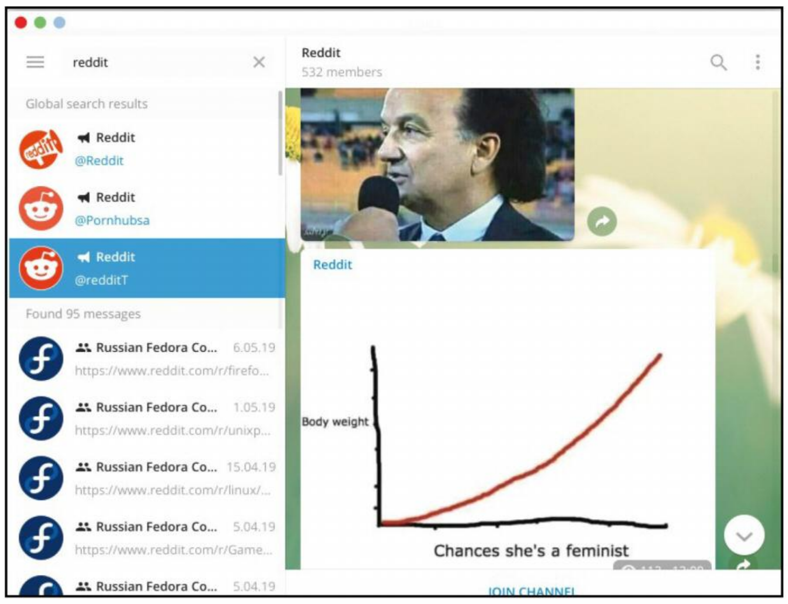
The online proofreader can only check text snippets below 20,000 words, which is still impressive. Apparently, the *LibreOffice* extension has no size limitations, but lacks support for 'n-grams' and is therefore limited in its grammar rules. To fix this, you're advised to manually download the 8GB of n-gram data and attach it to the application. The *LibreOffice* extension has a dedicated option for choosing the directory for n-grams, whereas command-line users should start the application with the `--languagemodel` parameter pointing to that directory. Using n-grams greatly improves grammar checking for English, German, French and Spanish by adding extra rules.

Despite the fact that *LanguageTool* offers paid plans for various extras, the quality of the open source version does not suffer, so everyone can enjoy this powerful grammar review tool and improve their linguistic competence. It doesn't end here, though; if you need to go deeper, be sure to take the neural-network version of *LanguageTool* (**https://github.com/gulp21/languagetool-neural-network**) for a spin.

TELEGRAM CLIENT

Kepka

Version: 2.0.0-rc2 **Web:** https://github.com/procxx/kepka



Kepka is your window to the world of great Telegram content worldwide.

Telegram is one of the most popular messengers in the world. Though its userbase is not nearly as large as those of *WhatsApp* or *WeChat*, it still boasts solid numbers and easily hits the top ten messenger list. The official *Telegram Desktop* app is open source, but we normally tend to avoid promoting it due to certain limitations, such as the custom-patched Qt libraries that it uses explicitly.

On the other hand, third-party *Telegram* apps such as *Cutegram* are often unable to compete with the official client in terms of features. The good news is that *Kepka* seems like a nearly ideal solution. It's a fork of *Telegram Desktop* with a nice set of fixes and improvements to the original, and we really like them.

Some features worth mentioning include the ability to compile the application against the official Qt packages, ditching unneeded GTK dependencies; *Clang* support; massive code clean-up and other under-the-bonnet things. From the user's perspective, *Kepka* is a fully-fledged *Telegram* client – it can do whatever its parent project can with no limitations.

Kepka respects your system's font settings and therefore looks at home on the desktop than *Telegram Desktop*. Its GitHub page has detailed information about compiling the code, with quick recipes for Arch and Fedora. Apart from having a bunch of build dependencies, the procedure comes down to `$ cmake . && make && sudo make install`, which isn't so scary.

Kepka can be used for texting your friends, discussing topics in groups or sharing your thoughts in channels. Channels make using *Telegram* and *Kepka* very social, but you'll likely need a web browser to join a channel for the first time, which is a bit of pain but not a disaster. Discover available channels at **https://tlgrm.eu/channels**. Make sure you associate **tg://** URLs in your web browser with *Kepka* to make the whole experience more fluid.

PASSWORD MANAGER

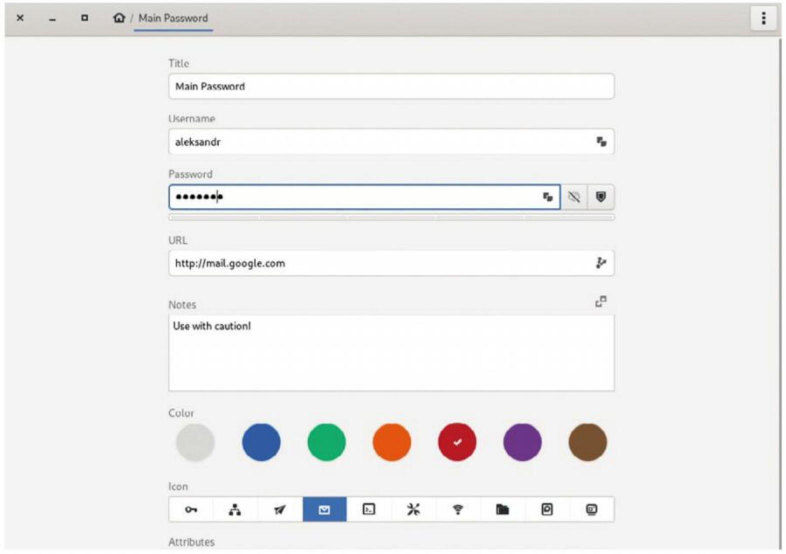
Password Safe

Version: 0.9.2 **Web:** https://github.com/FalkAlexander/PasswordSafe

There's always a trade-off between security and convenience, and that is perhaps why various password managers exist. Common sense suggests that as long as we deal with authentication almost every day, it's useful to store sensitive credentials in a dedicated vault. *Password Safe* can be a solid solution for that.

This is a small desktop application that stores your data in the *KeePass* v.4 format. Therefore, you can either start with an empty database, or import an existing **.kbdx** file from *KeePass* and continue using it with *Password Safe*. The application promotes itself as a Gnome-friendly frontend for *KeePass*, and is also known by the name *KeePassGTK*.

First-time users should press the '+' button in the top-left corner to create a new database file. After that, it is time to enter a new master password three times: two for setting it up and another one for unlocking the database. The logic behind *Password Safe* is exactly the same as the one behind other *KeePass* derivatives and frontends. The application lets you create entries and arrange them into groups. Each entry has a title, a username/password pair, an URL, a description field and an optional icon. All very simple, but don't miss those tiny buttons next to the field – they can copy the username or the password to the clipboard, follow the URL, or even generate a strong password for you.



Use the built-in randomness generator to further secure your account.

Password Safe supports databases that were set up for key files or composite authentication (key file plus password), although it doesn't provide any means for creating a key file from scratch. Still, the application is ideal for sharing a KBDX database with other *KeePass* frontends without compromising security.

Password Safe automatically locks your database in case of inactivity, so there's very little chance your sensitive data will leak. The easiest way to install *Password Safe* is to search for it on Flathub. Of course, if you aren't using Gnome you might as well stick with *KeePass* itself, but at least you have the option.

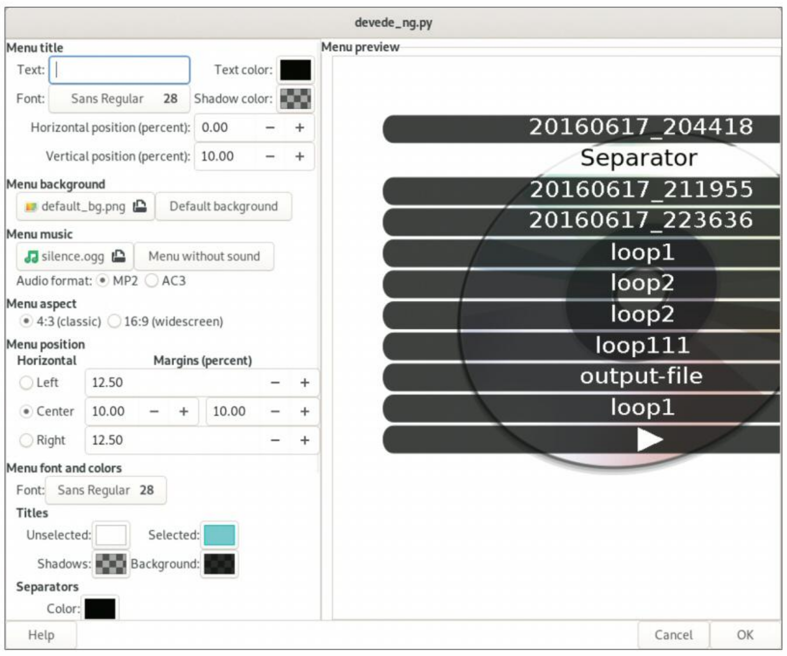
DVD AUTHORING

Devede NG

Version: 4.14 **Web:** https://gitlab.com/rastersoft/devedeng

This story has roots in the very distant days of *Linux Format*: the last time we took *Devede* for a test was in issue 110! Since then there have been many prophecies about the imminent disappearance of DVDs and the rise of web-based services, and although these predictions have partly come true, DVDs haven't disappeared entirely yet, and are still a useful medium. This means we still need an authoring tool like *Devede* – but if only it could be a bit more modern and keep up with the times...

Devede NG is the application that meets this need. It's a rewrite of *Devede* using Python3 and GTK3, and therefore offers a very similar set of features. Launching *Devede NG* brings up a welcome window with a selection of available types of discs to work on. If the list is greyed out, hit the 'Programs needed by *Devede*' button and check the list of third-party tools it needs to use. All of these are well-established FOSS tools such as *FFmpeg*, *AVConv*, *mkisofs* and *Brasero*. Once the new project is created, drag your media files onto the main area of the *Devede NG* window, arrange them and hit Next to proceed to disc burning prerequisites – it's all pretty simple stuff.



The configuration window for DVD menus is heaven for creative people.

After making sure that *Devede NG* implements all the beloved features of the original *Devede*, we found another point of interest under Edit > Preferences. The 'Multicore CPUs' drop-down menu offers several useful presets for fine-tuning performance. For instance, you can select the 'Use all except 1 core' variant to make sure that CPU-heavy operations such as encoding or converting will not render your system unusable.

No compiling is required to get it working, just launch the installer with:

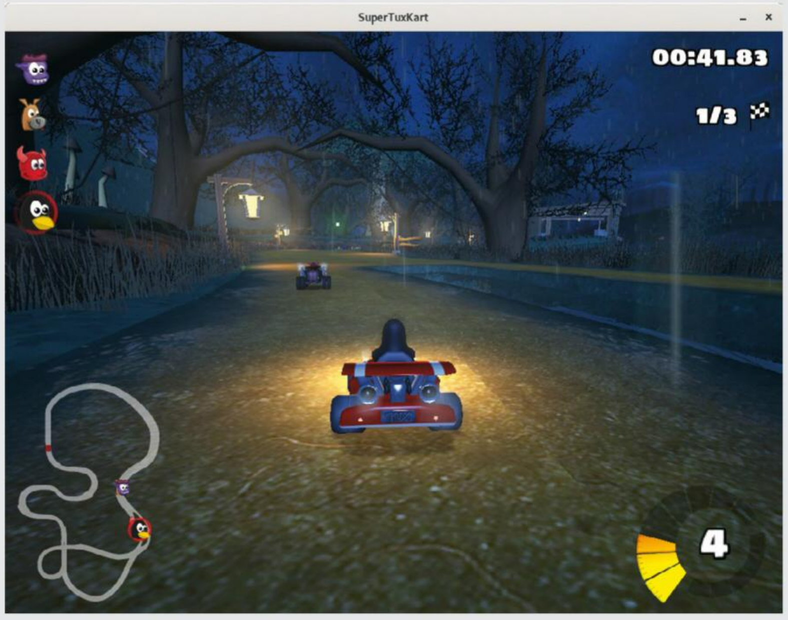
```
$ sudo ./setup.py install
```

This performs all required operations.

RACING GAME

SuperTuxKart

Version: 1.0
Web: https://supertuxkart.net



The picturesque scenery often prevents you from concentrating on the race.

SuperTuxKart 1.0 is here, and that's why we're busy choosing our favourite open source mascots and customising karts in order to see who's the best. This famed open-source racing game has greatly improved since we last saw it in **LXF232**. The biggest addition is the new LAN and online multiplayer functionality; this latest release finally lets you compete with other people.

In our humble opinion, the current LAN- and WAN-enabled game and the previous local-only version are like night and day. The online play feature in *SuperTuxKart 1.0* works just as you would expect: you need to select an existing server (or bring up your own) and wait for fellow racers to join in. Each server limits the maximum number of players, so you need to find one that isn't full, but which still has somebody other than you online. After that you may need to wait for the current game to end before you can participate in the next one, and then vote for the track of your choice. The actual track is selected based on the collective decision, which is nice and fair.

Real players are generally more skillful than *STK*'s bots on easy and normal difficulty levels. Struggling to outstrip an experienced player is hard, but in return it gives you much more of a thrill. As ever, *STK* has a generous selection of tracks, though some of them become accessible only if you complete others. The famous Ravenbridge Mansion, which was previously granted to *STK* donators only, is now available to everyone. Plus, there is the built-in add-on manager with a long list of community-contributed tracks and arenas.

There's a definite increase in *STK*'s online mode activity these days, so take care of your ping when joining an online team. Figures around 100ms are fine, whereas pings of ~300ms will cause higher latencies and things like sudden kart stuttering. To avoid that, try to choose players near you.

ECONOMIC SIMULATOR

OpenPatrician

Version: 1.0.1 **Web:** https://sourceforge.net/projects/openpatrician

Trading in the Hanse of the 14th century is no less thrilling than the one in the current-day EU.

The late middle ages in Northern Europe and the Baltic region were a fast-changing time, when landlords started to lose their power due to the powerful Hanse trade union. In *OpenPatrician*, you are a young trader who buys and sells goods in the network of Hanseatic towns. At the beginning of the game you choose your name, a home town and finally start with one trade ship at your command.

OpenPatrician is not turn-based; it's constantly running in in-game time, which is important to remember. For instance, taking money as a loan implies payback in a given period of time, so it's unwise to be idle when you owe money.

If you haven't played this before, we strongly recommend exploring the documentation for *The Patrician* (1993), a proprietary German game from which *OpenPatrician* took its inspiration. The game is fully controlled by mouse, and it offers plenty of places and objects that you can click on and communicate with. *OpenPatrician* is a captivating and complex economic simulator with lots of details a player should keep in mind.

There are two main goals in the game: one for earning profit by selling goods to other towns, and another for advancing your own career position. The latter is no less important than the economic part. Your influence grows if you become a mayor of a town, but the most powerful position is certainly the Äldermann, a head of the whole Hanse.

In general, the list of your expenses is much wider than the list of earnings. You need to pay a salary for your workers, keep your fleet in a good shape, bribe other mayors and even arm your ships with ballistas and catapults in order to hold pirates off at gunpoint. Lots of scenarios are possible in *OpenPatrician*, including hiring a trade manager, who will perform buy and sell operations for you. It's an incredible historic simulator that is worth getting used to. Fire up OpenJDK 11 or newer in order to play the Java-based *OpenPatrician*, and travel back in time!

SYSTEM CLEANER

Stacer

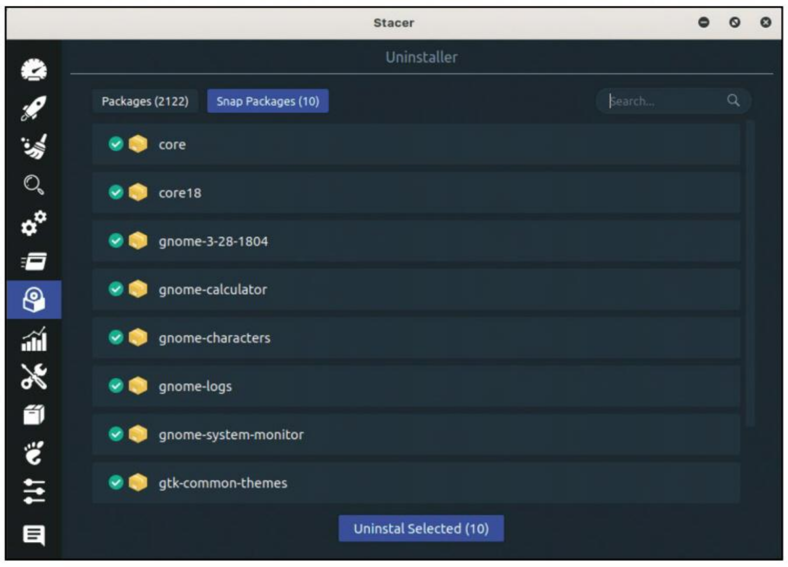
Version: 1.1.0 **Web:** https://github.com/oguzhaninan/Stacer

The niche of housekeeping applications for Linux seems to be ripe in potential. Right now there aren't many such apps that immediately come to mind, apart from KDE's *Sweeper* and the built-in cleaner in elementaryOS. *Stacer* is much more powerful than those small tools; in fact it claims to be an all-in-one universal solution for cleaning unneeded files, managing apps and tweaking desktop settings.

We took the older *Stacer* version for a spin in **LXF237**, but since then the application has drastically improved and is equipped with lots of extra features. This new release takes *Stacer* to new heights with the support for uninstalling Snap packages, a built-in file search tool, a disk usage meter and hosts manager. *Stacer* is a first-class citizen in Ubuntu and Ubuntu-based distributions thanks to specific features for managing Snaps, *Apt* and GNOME/GTK settings. When used with other Linux flavours, all inapplicable features are gracefully hidden.

The newly added functionality is what we enjoyed most in *Stacer 1.1.0*. The ability to remove several Snap apps in bulk, search for files and alter the contents of **/etc/hosts** without leaving the great-looking interface is surely a sane design decision. The Gnome section mostly offers the same switchers as *Gnome Tweaks* (just reordered), but there are some unique things such as hardware-accelerated text rendering quality, desktop icon enablers and advanced window-manager tweaks. The core *Stacer* mission remains the same, however: use this app to detect and eliminate redundant and outdated cache files from your machine.

Stacer was an Electron app in its early days, but in recent years it's become C++ and Qt5-based software with a handy universal Appimage suggested as the main download. The Appimage bundle is supposed to run on any modern Linux distribution – just don't forget to make the file executable first.



Got too many Snaps you want to remove all at once? Stacer will save you time!

IMAGE GENERATOR

WallGen

Version: GIT **Web:** https://github.com/SubhrajitPrusty/wallgen

Search for 'Ubuntu wallpaper creation' on the web and you'll get lots of results about wallpaper changing, making a slideshow, customising images and other great and useful content. But there's unlikely to be anything related to the design process of those simple yet appealing abstract lines and gradients. *WallGen* is a superb Python3-based image generator that can create abstract polygonal patterns of the kind beloved by the Ubuntu team.

It's a command-line utility that can make wonders out of nothing, taking only few text arguments for input. The output usually looks like a shape-based pattern or a randomly filled surface, depending on the conditions you provide to *WallGen*.

Download the code, *cd* to the main directory and install *WallGen* using the following command:

```
$ pip install --editable .
```

By default, Python applications installed for a non-root user should reside in **$HOME/bin** or **$HOME/.local/bin**. Now that you have *WallGen* installed, let's go ahead and generate some eye-candy:

```
$ wallgen poly 1000 --colors "#ff0000" --colors "#00ddff"
```

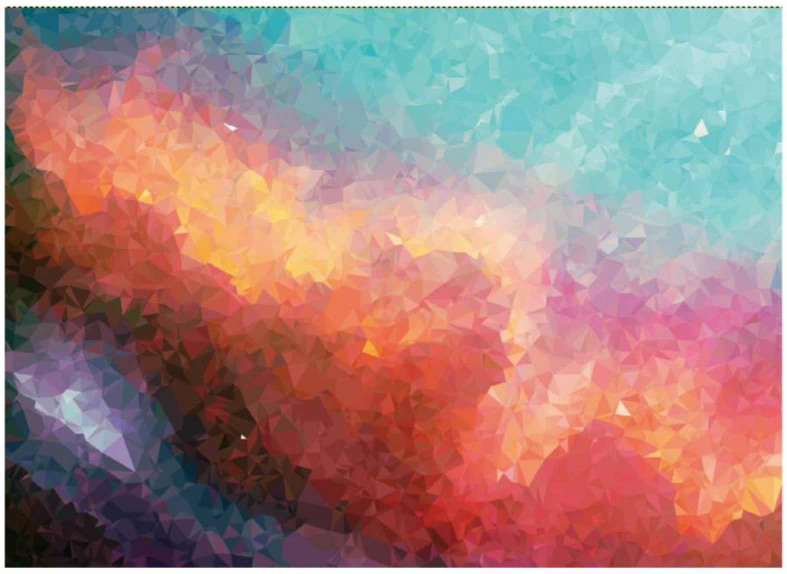
The above command creates a polygonal net made up of 1,000 nodes and coloured with a smooth red-to-aquamarine gradient. The more nodes you request, the larger the resulting image. *WallGen* is capable of generating ordered patterns as well:

```
$ wallgen shape 2000 -t hex -c "#ff0099" -c "#00ddff"
```

That will form something like a coloured honeycomb. Change `hex` to `square`, `diamond` or `triangle` for more patterns. But the greatest feature of *WallGen* is its ability to process existing images:

```
$ wallgen pic poly /path/to/image -p 50000
```

Turn an ordinary photo or drawing into a piece of abstract art! LXF



Use Wallgen to make great-looking backgrounds, and run away with any wallpaper contest!

PYTHON/GO

Parsing XML files

The code
At linuxformat.com/archives and on the LXFDVD

John Schwartzman shows how to parse and display XML documents using code written in Python and Go.

OUR EXPERT

John Schwartzman is a software engineering consultant to business and government. He also teaches computer science at a local college.

One of the easiest ways to learn a new programming language is to build a program first in a familiar language (Python) and then convert it to the language you want to learn (Go). That makes it easy to compare the features and idiosyncrasies of the two programming languages.

Figure 1 *(right)* shows the output of our Python program when reading an XML file. The Python parser, **XMLParse.py**, doesn't have the ability to display comments in XML files. It knows how to ignore comments (after all, comments are for people, not programs), but not how to display them. Figure 2 *(page 93)* shows the output of our Go program, **XMLParse.go**, on the exact same XML file, **postfix.xml**. The Go version clearly does know how to display comments.

Let's start by looking at **XMLParse.py**. Create a working directory and place **XMLParse.py** there along with the sample XML files, **addressbook.xml** and **postfix.xml**. Make sure that your version of **XMLParse.py** is executable. If it's not, use the alias `mx (alias mx='chmod +x')` to make it executable for everyone.

```
mx xmlParse.py
./xmlParse.py postfix.xml
```

In the main section, which every program must have, we first look at the command line arguments to check whether the user has entered the correct number of arguments and whether the user wants help. If so, we invoke the `usage()` function with an error code of 0, which tells the system the we exited normally. If the user has entered the wrong number of command line arguments, we invoke the `usage()` function with an error code of 1 to indicate to the system that an error occurred. If we pass through this error checking successfully we invoke `main()` with `sys.argv[1]`, which is the filename of the XML file we want to view. `sys.argv[0]` is, of course, the program file path for XMLParse.

The `main()` function opens the specified XML file, instantiates an `XmlContentHandler` class object and initialises it with the open file. Since these are actions that could conceivably fail, we wrap them in a `try/except` block to handle potential errors. These blocks are similar to the `try/catch` blocks in C++, Java and C#. One of the programmer's most important jobs is to try to anticipate potential errors and to let the user exit from them gracefully.

```
def usage(exitFlag):
  print ('\nUSAGE: ./xmlParse.py xmlFileToView\n\n')
```

QUICK TIP

A code editor such as *Microsoft Code* that 'knows' Python and Go will save you a lot of time and a lot of pain. It will perform code completion and it will tell you what is and what is not syntactically correct for each programming language.

Figure 1: Our program XMLParse.py displaying postfix.xml.

```
  sys.exit(exitFlag)

def main(sourceFileName):
  try:
    source = open(sourceFileName)
              # instantiate and initialize
XmlContentHandler
    xml.sax.parse(source, XmlContentHandler())
  except xml.sax.SAXParseException as e:  # handle
parser error
    print('\nFailed to parse ' + sourceFileName +
        '. This appears not to be a valid xml document: ' +
        e.getMessage() + '\n\n')
    sys.exit(2)
  except OSError as e:          # handle os error
    print('\nFailed to open ' + sourceFileName + ': ' +
str(e) + '\n\n')
    sys.exit(3)
if __name__ == "__main__":
  if sys.argv[1] == "-h" or sys.argv[1] == "--help":   # does
user want help?
    usage(0)
  if len(sys.argv) != 2:   # there must be 2 arguments
    usage(1)
  main(sys.argv[1])
  print('')
  sys.exit(0)
```

The `XmlContentHandler` (which we instantiate and invoke in `main`) walks through the XML file and invokes functions that correspond to the artifacts found in the file. When the handler encounters a startElement, which looks like <elementName>, it invokes the `XmlContentHandler` member function `startElement()`. When it encounters an endElement, which looks like </elementName>, it invokes the `XmlContentHandler` member function `endElement()`.

The `startElement()` member function also handles element attributes. When we encounter string characters, we invoke the `characters()` member function, which pushes the characters into `charBuffer`. This buffer contains element data and it is accessed and printed when the handler encounters an endElement. Each time the handler encounters a startElement, it pushes the name onto the `elementStack` and uses the size of the stack to determine where on the X axis it should print the element name. At that point, it increments the row and adds spaces to indicate the column position where the element name should be printed.

```
def startElement(self, name, attributes):          # we've
encountered a startElement
  pos = self.pushElementToStack(name)
  self.writeNewLine()
  self.writeSpaces(pos, '   ')   # write 3 spaces per index
  self.writeStartName(name)
  self.writeAttributes(attributes)
  self.nLastWritePos = pos
```

A stack is a last in, first out (LIFO) data structure. We are constructing our stack with a list data structure. When we push a name onto the stack it adds the name to the top of the stack, which is the end of the list. When we pop a name from the stack, it removes the name from the top of the stack. The depth of a name in the stack determines where on the X axis we're going to print startElement and endElement. Elements that are deeply nested will be further to the right than less deeply nested elements. The name of the first startElement will be placed on the stack with a depth of 0. It should appear on the left edge of the display (X position = 0). The last endElement in the file (which must correspond to the first startElement) should also have a depth of 0. It should also appear on the left edge of the display.

Each time the handler encounters an endElement, it pops the name from `elementStack`. It then checks to see if there is data associated with the element. If so, it prints the data. It may then add spaces to change the column position where the closing element name should be printed. If we've just written in the same position as the depth of element on the stack, we write in the current X position. Otherwise, we're in a new Y position and we use the depth of the element on the stack to indent the closing element name.

```
def endElement(self, name):              # we've
encountered an endElement
  pos = self.popElementFromStack(name)
  charStr = self.getCharacterData()
  self.writeElementData(charStr)         # write it
  if pos < self.nLastWritePos:           # write name at
current x pos?
    self.writeSpaces(pos, '   ')  # position endElement on
x-axis
  self.writeEndName(name)
```

That's pretty much all there is to the Python program. When the parser runs out of elements to process, the program returns to `main()`, which exits with a return code of 0 to indicate success.

The Go program works exactly like the Python program. Let's first look at some of the obvious differences between the two programs. In Go programs, functions and constants may start with a capital letter

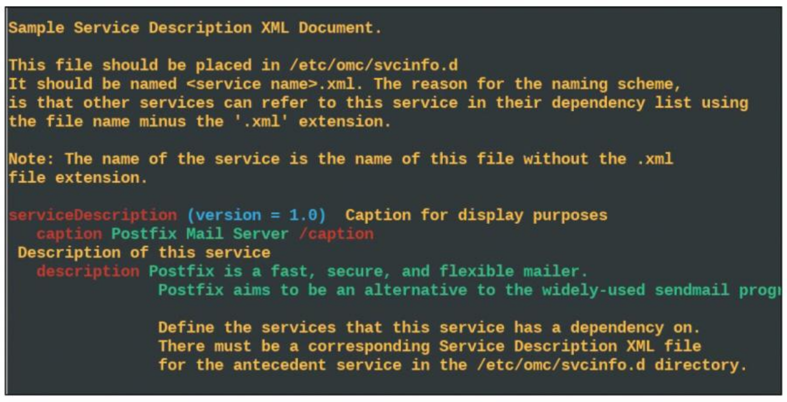


Figure 2: XMLParse.go displaying the same postfix.xml.

only if you intend to export them. Since we are using them locally, inside the main package, they must start with a lower-case letter.

In Python, we can delimit strings using single quotes or double quotes. In Go, double quotes must be used for strings and single quotes must be used for characters. Note also that functions in Go are delimited with curly braces `{}` rather than a colon and a new indented tab position, as in Python.

Further, the opening curly brace must appear on the same line as the function definition. You are forced into a Kernighan and Ritchie programming style (as in the authors of the original C) whether you like it or not. There are no semicolons at the end of statements in Go as there are in C and C++. They are injected automatically and invisibly by the Go compiler.

Compare the constant declarations in the Python program and in the Go program. In the Go program they are lower-case, and strings are formed by concatenating characters using the double quote symbol. Even the comments are different. In Python, we use the `#` symbol to delimit comments, as we do in other scripting languages. Like C and C++ programs, Go uses the `//` comment delimiter and also the `/* this is a comment */` notation.

QUICK TIP

Your distro's package manager should have Go available. Sometimes it's known as golang. The latest version at this time is 1.12, so look for go1.12, go1.12-doc and go1.12-race. Of course, if there's a later version, get the latest version that you can find. It's also available at https://golang.org/dl.

» XML FILES

There are many XML files in your Linux Distro. There are 3,068 in **/usr/share**, and 8,630 overall in our SUSE Tumbleweed distro. (Mint 19 contains 2,350 in **/usr/share** and 21,473 overall!) Many of these files contain data, configuration information and translation information, and are used by programs on your system. These XML files are marginally readable by humans.

Notice that the XML information is organised by elements. In **postfix.xml**, for example, we start with 'serviceDescription' which has an attribute name/value pair, 'version = 1.0', associated with it. Then we have the element 'caption', inside the element 'serviceDescription'. The caption element has the data 'Postfix Mail Server' associated with it. Then another new element, 'description', also containing data. Then comes the element 'dependsOn', which has two elements inside of it: 'serviceName = network' and 'serviceName = syslog'. Notice that these two elements are followed by their 'end' elements followed by the 'end' element for 'dependsOn'. We then have the elements 'startCommand', 'reStartCommand', 'stopCommand' and 'statusCommand' at the same level as the 'dependsOn' element. After the 'end' element for 'statusCommand', we have the 'end' element for 'serviceDescription'. At this point the root element, 'serviceDescription', has been closed and that marks the end of the file.

Take a look at **postfix.xml** with the *less* command and verify that you can see all of this information inside of it.

»

QUICK TIP

A very common mistake is to build your Go program using the command 'go build XMLParse.go' and then type 'XMLParse' to run it. The *XMLParse* executable is in your working directory, but you can only run it by typing ./xmlParse filename. Typing XMLParse will find the XMLParse that lives on your PATH.

Go programs can be interpreted or compiled. When you are starting a project, you use the interpreter by invoking `go run XMLParse.go XMLFileName` at the command line. To use the compiler, you invoke `go build XMLParse.go`, which creates the executable program *XMLParse* in your working directory. You then run the compiled version by invoking `./XMLParse XMLFileName`. Go programs don't have classes per se, but they do have structs. Functions can be added to structs in the same way that functions can be added to classes in a pure object-orientated language.

Create a working directory and place **XMLParse.go** there along with the sample XML files **addressbook.xml** and **postfix.xml**. Compile the program and copy it to your **~/bin** directory:

```
go build xmlParse.go
cp xmlParse ~/bin
xmlParse postfix.xml
```

Let's begin our exploration of the Go version of *XMLParse* with the `main()` function. In Go, `main()` takes no arguments and does not return a value. In `main()` we check that the user has entered two arguments and then determine whether the user wants help. In either case we invoke the `usage()` function with an exit code. This function is at the same level as the `main()` function and all of the other functions. Functions don't take a class instance as an argument as they do in Python. The functions are much simpler. Notice also how all of the action takes place in `main()`.

After checking the run-time arguments, we try to open the designated XML file for reading. Note that Go does not have the equivalent of `try/except` error handling. You handle errors by returning two separate values from a function call that could result in an error. For example, in the main function we have:

```
xmlFile, e := os.Open(os.Args[1])        // os.Args[1] is the
xml file to view
if e != nill {
  fmt.Printf("\nProblem reading %s:\n\n", os.Args[1], e)
  os.Exit(2)
}
```

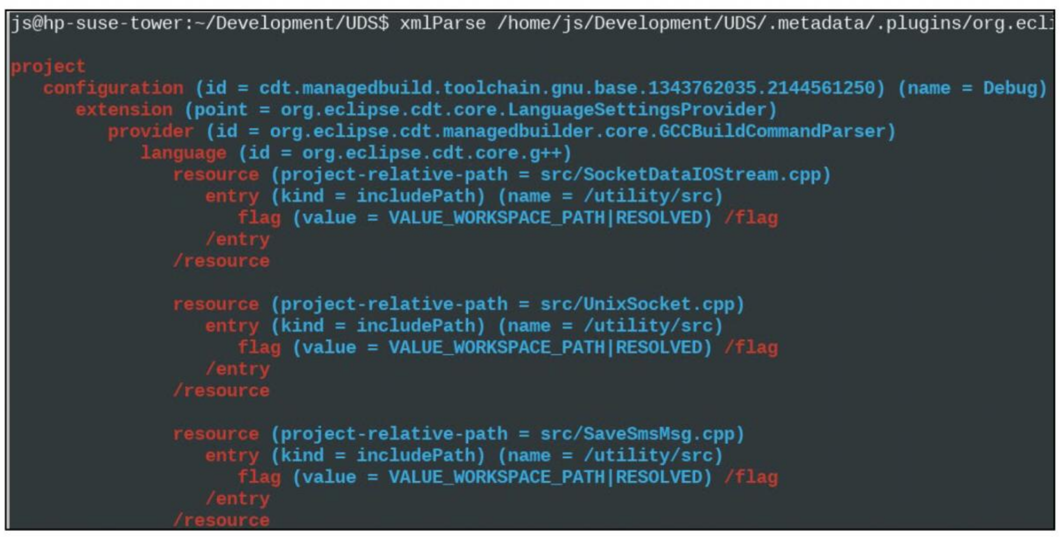
`os.Open()` returns a file object and possibly an error if it couldn't open the XML file. If you didn't want to handle errors on your first attempt through the code, you could use the dummy variable `_` and simply let potential errors blow up the program. You can get away with this if it's you running it. If you're writing programs for others to use, you have to be more careful. The quick and dirty method would look like this:

```
xmlFile, _ := os.Open(os.Args[1]
```

Next, we instantiate `xml.NewDecoder()` and pass the open XML file to it for processing. Then we initialise the empty `elementStack`, which we'll use in the same way that we did in the Python program. We now enter a `for` loop where we check each token found in the XML file by the `NewDecoder` object. In this loop we wait for tokens as `xml.NewDecoder()` processes the XML file.

```
for { //while there are tokens, stay in for loop
 // get a new token
 t. err := decoder.Token()
 if err != nil && err.Error() != "EOF" {
  fmt.Printf("Error: %s\n", err)
 }
 if t == nil {
  // we've reached the end of the xml document
  break  // exit the for loop
 }
 // Inspect the type of the token
 switch se := t.(type) {
 case xmlStartElement:      // we've encountered a
start elements
  pos := push(elementStack, se.Name.Local) // push it
onto the stack
  writeNewLine()
  writeSpaces(pos, "   ")      // write three spaces per
index position
  writeStartName(se.Name.Local)
  for _, a := range se.Attr {
   writeAttribute(a.Name.Local, a.Value)
  }
  nLastWritePos = pos
 case xml.EndElement:       // ooh an endElement
  pos := pop(elementStack, se.Name.Local) // pop it
  if pos < nLastWritePos {   // write name at x pos?
   writeSpaces(pos, "   ")   // set position end element
  }
  writeEndName(se.Name.Local)
 case xml.CharData:         // element data
  // remove any surrounding whitespace
  data := strings.TrimSpace(string(t.(xml.CharData)))
  if data != "" {
   writeCharacterData(data) // write it at current x,y
 }

 case xml.Comment:          // comments
  data := string(t.(xml.Comment)) // write it at x,y
  writeComment(data)

 case xml.Directive:  // we've encountered a directive
  data := string(t.(xml.Directive))
  writeDirective(data)

 }          // end of switch statements
}           // end of for loop

fmt.Printf(white)  // rest ore normal screen formatting
writeNewLine()
}
```

In this way, we process each of the tokens we encounter in the XML file. The only difference between the Python parser and the Go parser is that Go recognises comments and directives and it delivers element data all at once. We respond to each of the tokens we encounter in almost the same way we do with the Python parser. Notice that there are no `break` statements in the `switch` statement – Go inserts them



```
js@hp-suse-tower:~/Development/UDS$ xmlParse /home/js/Development/UDS/.metadata/.plugins/org.ecl

project
   configuration (id = cdt.managedbuild.toolchain.gnu.base.1343762035.2144561250) (name = Debug)
      extension (point = org.eclipse.cdt.core.LanguageSettingsProvider)
         provider (id = org.eclipse.cdt.managedbuilder.core.GCCBuildCommandParser)
            language (id = org.eclipse.cdt.core.g++)
               resource (project-relative-path = src/SocketDataIOStream.cpp)
                  entry (kind = includePath) (name = /utility/src)
                     flag (value = VALUE_WORKSPACE_PATH|RESOLVED) /flag
                  /entry
               /resource

               resource (project-relative-path = src/UnixSocket.cpp)
                  entry (kind = includePath) (name = /utility/src)
                     flag (value = VALUE_WORKSPACE_PATH|RESOLVED) /flag
                  /entry
               /resource

               resource (project-relative-path = src/SaveSmsMsg.cpp)
                  entry (kind = includePath) (name = /utility/src)
                     flag (value = VALUE_WORKSPACE_PATH|RESOLVED) /flag
                  /entry
               /resource
```

Figure 3. XMLParse.go displaying an Eclipse XML file that indicates the files in a C++ project.

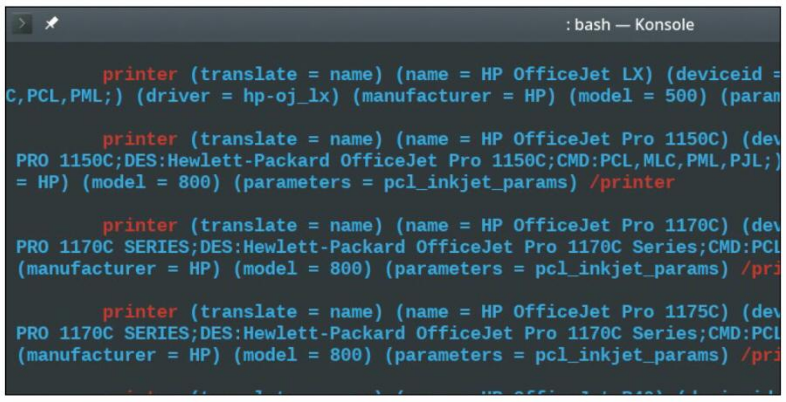


Figure 4. XMLParse.go displaying all the printers that will populate a dialogue box.

automatically and invisibly. The `for` statement is the only looping construct in Go. Notice that in the function `writeSpaces()` we use `i:=` for the initial assignment to i, followed by the condition to check, followed by the action to take on each iteration. These are separated by semicolons.

```
func writeSpaces(pos int, chars string) { // position the cursor column
  for i := 0; i < pos; i++ {
    fmt.Printf(spaces, chars)
  }
}
```

The constant `spaces` is equal to `black + "%s" + white`. The constant `black` makes text black on a black background, or invisible. We print `pos` copies of `" "` on the screen and return to our default colour assignment, `white`, which is white text on a black background. `writeSpaces()` is how we position the cursor where we want it on the X axis.

Notice, also, that all of the `const` colour macros end with `white`. This is so that if the program breaks, your terminal should be restored to a normal condition and you should be able to see anything you type on the console. Compare the Python version of `writeSpaces()` to the Go version. Notice that in the Python version, we iterate over the `range()` function and use a dummy variable for the index.

We now turn to the functions that are invoked when the Go parser reaches an xml.startElement or an xml.endElement token.

```
func push(s *list.List, name string) int {
  pos := s.Len()      // use the index before push
  s.PushBack(name)           // push it onto the stack
  return pos
}
func pop(s *list.List, name string) int {
  e := s.Back()       // get the last element in the list
  if e.Value == name {
    s.Remove(e)       // pop it from the stack
  } else {
    fmt.Printf("%s\nError: %s was not at the top of the stack.\n\n",
                              white, name)
    os.Exit(4)
  }
  return s.Len()      // use the index after pop
```

These functions are designed to use the `list.List` structure that we imported from the `container/list` package. We specify that these functions take a pointer to a list (`s *list.List`) and a string (`name string`) as arguments, and return an `int` which corresponds to the depth of the stack. Note the function signatures for `push` and `pop`; they tell us the names and types of the arguments and the type of the return value. This form of function signature is unique to Go. What does it mean to pass a pointer to a List? It means that we're passing the address of the List. We don't want to pass the List around; it's too big. Passing the address of the List is more economical.

When the parser encounters an xml.startElement, we push the element name onto `elementStack`. When the parser encounters an xml.endElement, we pop the element name off of `elementStack`. The depth of the stack tells us where on the X axis we should print the element name. After the push we move to a new line and call `writeSpaces(pos, " ")`, where `pos` is the value returned from the `push` function. `writeSpaces()` indents three spaces for every level that the element name is nested on the stack. In this way we try to print the startElement name and the endElement name at the same horizontal position. Notice that we don't always print an endElement name at the same horizontal position as a startElement name. When it's written on the same line as attributes or data we print it at the current X position.

Figure 3 *(see far left)* shows an XML file used by *Eclipse CDT* to keep track of project source file members. Note that the attributes of the resource elements each specify a C++ program in the project path. *Eclipse* uses this and other XML files to automatically generate *make* files for your C/C++ projects. Similarly, when you add a new printer or printer driver to your system, a dialogue box is populated with a long list of printers. That dialogue is populated from an XML file *(see Figure 4 above)*.

XML documents are used to store data, configuration information, translations, really almost anything. We've used XML files to update a database using Java and Hibernate. Grab a few assorted XML files from your distro and try to view them using *XMLParse* – but beware, not every XML file is well-formed. There are lots of files in your distro that are not. *XMLParse* will spit them out if they don't follow the rules, while the programs that use the XML files may be more forgiving. Our two parsers are simply general-purpose XML file viewers that organise and colour-code the contents of XML documents. LXF

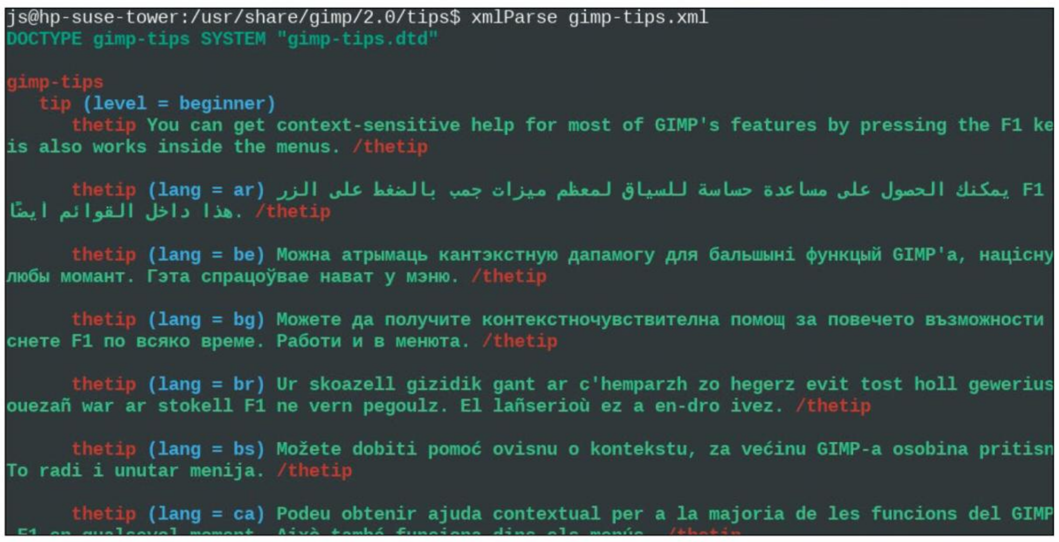


Figure 5: XMLParse.go displaying Gimp tool-tip translations.

KAFKA

Build utilities with the Kafka server

Mihalis Tsoukalos explains how to install and create utilities that use Apache Kafka, with its "producer" and "consumers" stream model.

OUR EXPERT

Mihalis Tsoukalos is a UNIX person and the author of *Go Systems Programming* and *Mastering Go*. You can reach him at **https://www.mtsoukalos.eu** and @mactsouk.

Nowadays, quite a few data architectures involve both a database and Apache Kafka, which is a distributed streaming platform and the subject of this tutorial. You can also find Kafka described as a publish-subscribe message system, which is a fancy way of saying the same thing as it being a distributed streaming platform. As Kafka is optimised for speed, it doesn't offer too many facilities to users or administrators – like the ones offered by relational databases – and Kafka commands are long and difficult to remember, worried yet?

Don't worry as here we'll look at the basics of Kafka, including plenty of the commands and two Go utilities for working with it.

Installation

We'll use Kafka from a Docker image. The main reason for this is that Kafka has many dependencies and parameters that complicate the installation process – but Docker images come ready to run and include all required dependencies.

For this tutorial we'll use a Kafka Docker image offered by Landoop. This is one of the best that you can find because it enables you to work with Kafka as soon as you download it! Other Kafka Docker images, including the one offered by Confluent, require many customisations that demand time, which might discourage amateur users.

First, you will need to download the Kafka Docker image by executing the following command:

```
$ docker pull landoop/fast-data-dev:latest
```

If you ever want to update that image to the latest version, you can simply rerun the `docker pull` command. Next, you will need to run the Kafka Docker image you downloaded in order to be able to connect to Kafka, its services and its ports:

```
$ docker run --rm --name=kafka-box -it -p 2181:2181 -p 3030:3030 -p 8081:8081 -p 8082:8082 -p 8083:8083 -p 9092:9092 -p 9581:9581 -p 9582:9582 -p 9583:9583 -p 9584:9584 -e ADV_HOST=127.0.0.1 landoop/fast-data-dev:latest
```

This command makes Kafka and the desired port numbers available to the world; put simply, you can

QUICK TIP

You can learn more about Kafka by visiting https://kafka.apache.org and https://docs.confluent.io/current/

Figure 1: How you can list the topics in a Kafka server that runs on a Docker image in two ways, as well as how to find the version of the Kafka server you are using.

connect to some Kafka-related services from your local machine without the need to connect to the Docker image first. Kafka Broker listens to port number 9092, whereas Zookeeper listens to port number 2181. Last, the Schema Registry listens to port number 8081 and Kafka Connect to port number 8083.

Because this Docker image is part of a bigger Landoop project, you can visit **http://127.0.0.1:3030** after running it to display lots of handy information that will make your life much easier.

The following commands show how to start a UNIX shell, which in this case is *bash*, in the Docker image:

```
$ docker exec -it kafka-box bash
```

Note that this command connects to a running Docker image named `kafka-box`, which was defined as a parameter of the `docker run` command. If you used a different name, you will need to change that string for the command to work.

The only disadvantage of using a Docker image is that after you shut down the Docker image, data and changes will be lost. The good thing is that there are workarounds to this problem; however, talking about them is beyond the scope of this tutorial. The biggest advantage of using a Docker image is that everything you do to that image, including deleting things and stopping services, will not affect the image itself and your Linux machine. Thus, if you think there's something wrong with the commands you've executed you can simply stop the running Docker image and restart a new one. If you really need data persistence and you aren't familiar with *Docker*, you can always install Kafka on your Linux machine using your favourite package manager or download it and install it by following the instructions found at **https://kafka.apache.org/quickstart**.

You can list the available Kafka topics from your UNIX shell as follows:

```
$ docker run --rm -it --net=host landoop/fast-data-dev kafka-topics --zookeeper localhost:2181 --list
```

This command can also be executed while being connected to the *bash* shell of the Docker image:

```
$ kafka-topics --zookeeper localhost:2181 --list
```

You can find the version of Kafka you're using by executing `kafka-topics --version` from within the Docker image. Figure 1 *(see page 88)* shows the output of some of the commands presented in this section. As you can see, the default Kafka installation of the Docker image that we used comes with many Kafka topics. The next section will show you how to create your own Kafka topics and insert your own data in them.

Adding data to Kafka

For the presented commands to work you will need to be connected to a *bash* shell in the Docker image, or have Kafka installed on your Linux machine. You can create a new Kafka topic named `NEW-LXF-TOPIC`, with three partitions, as follows:

```
$ kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic NEW-LXF-TOPIC
```

The command might look a little complex at first but this is how Kafka works. You will have to connect to the Zookeeper server and you will have to define all required parameters as command line arguments. You can increase the partitions of an existing topic as follows:

```
$ kafka-topics --zookeeper localhost:2181 --alter --topic NEW-LXF-TOPIC --partitions 16
```

However, if the new value is smaller than the old value, the number of partitions will not be reduced. You can insert data into a Kafka topic from your Linux shell as follows:

```
$ kafka-console-producer --broker-list localhost:9092 --topic NEW-LXF-TOPIC
```

After you execute this command, you will have to type the data you want. Please bear in mind that all records should have the same format, based on the format of the first record that has been written to a Kafka topic – consistency is key here because otherwise you will have difficulties and failures. Count the number of records that exist in all partitions of a Kafka topic as follows:

```
$ kafka-run-class kafka.tools.GetOffsetShell --broker-list localhost:9092 --time -1 --topic NEW-LXF-TOPIC | awk -F: '{print $3}' | awk -F: 'BEGIN{sum=0} {sum+=$1} END {print sum}'
```

View that data in a topic on your screen as follows:

```
$ kafka-console-consumer --from-beginning --bootstrap-server localhost:9092 --topic NEW-LXF-TOPIC
```

Note that the aforementioned command will present

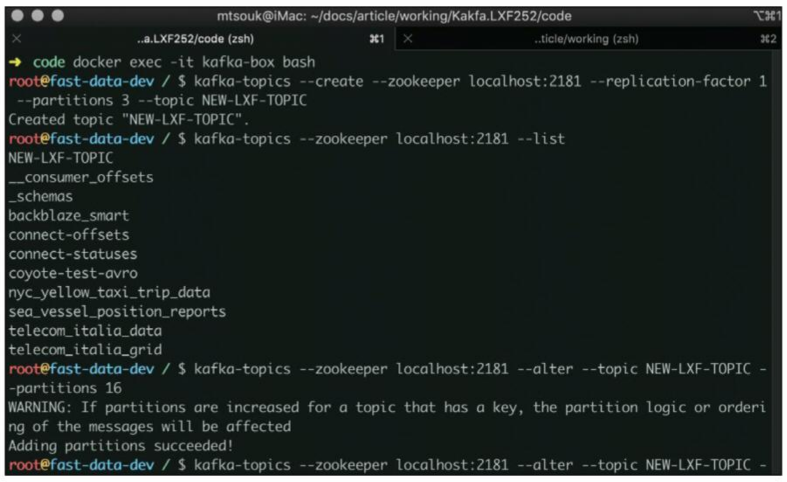


Figure 2: The use of some basic Kafka commands that enable you to list all available topics, define a new topic with the desired number of partitions, write data to a topic and read from a topic.

QUICK TIP

Kafka 'connectors' can be either 'sources' or 'sinks'. Sources enable you to import (producer) data to Kafka, whereas sinks enable you to export (consumer) data. You can see a list of supported connectors at www.confluent.io/hub.

the entire dataset from the selected topic and will keep waiting for new data; this means that you have to manually end the command by pressing Control+C. If you just want to display new data, you should omit the `--from-beginning` parameter from the `kafka-console-consumer` command. Lastly, if the topic is empty, you will see no output until you add records to that topic. Avoid manually terminating `kafka-console-consumer`; you can use the `--timeout-ms` parameter to define a time-out period of waiting for new data as follows:

```
$ kafka-console-consumer --from-beginning --bootstrap-server localhost:9092 --topic NEW-LXF-TOPIC --timeout-ms 100
```

Delete all data from a Kafka topic without deleting the topic itself as follows:

```
$ kafka-configs --zookeeper localhost:2181 --entity-type topics --alter --add-config retention.ms=1000 --entity-name NEW-LXF-TOPIC
$ kafka-configs --zookeeper localhost:2181 --entity-type topics --describe --entity-name NEW-LXF-TOPIC
$ kafka-configs --zookeeper localhost:2181 --alter --entity-type topics --entity-name NEW-LXF-TOPIC --delete-config retention.ms
```

» BASIC CONCEPTS

A messaging system such as Kafka enables you to send messages between processes, applications and servers. Applications connect to Kafka to send or get data. Strictly speaking, a Kafka 'topic' is a unit of storage in Kafka: data in Kafka is stored in topics. A topic is shared, hence you have a topic partition. Therefore, you can think of a Kafka topic-partition tuple as an append log. Simplistically and without considering partitions, you can think of a Kafka topic as a table in a relational database.

A Kafka 'broker' is a Kafka server. Many Kafka brokers create a Kafka cluster. A Kafka 'producer' is a program that writes data to a Kafka topic, whereas a Kafka 'consumer' is a program that reads data from Kafka. A 'partition' is the smaller part of a Kafka topic. A Kafka topic consists of one or more partitions – this mainly depends on the amount of data that you have to deal with. It is better for topics with huge amounts of data to have multiple partitions. Among other things, this enables multiple consumers to read from the same topic in parallel. Additionally, a topic can be split across multiple brokers at the partition level. Replication in Kafka happens at the partition level. Each record in a partition is separated from the other records using an offset. As Kafka knows nothing about the format of a record, the offset is really important information. Last, Zookeeper (**https://zookeeper.apache.org**) is used for doing the housekeeping of Kafka by containing configuration, naming and synchronisation information. This means that you cannot run Kafka without Zookeeper.

»

QUICK TIP
If you are really into Kafka and you want to make the experience as smooth as possible, you should try Lenses (https://docs.lenses.io), a product from Landoop (www.landoop.com). Lenses Box is the free yet fully featured version of Lenses.

The aforementioned commands need a little explaining because they don't actually delete any data: Kafka purges data from topics. In order to purge all data from the Kafka topic, you need to change the retention time of that topic. The default retention time is seven days, so you need to change the retention time to one second, after which the messages from that topic will be deleted automatically. Then you can change the retention time of the topic back to seven days by deleting the existing value.

Should you wish to keep these messages longer, it would be better to create a Kafka consumer and store them to another database or place. Figure 2 *(see page 89)* shows the output of some of the commands presented in this section. Try adding and deleting topics and data to Kafka on your own.

Basic administration

It's essential to know a few basic commands that enable you to administer a Kafka server. First we'll learn how to delete an existing Kafka topic:

```
$ kafka-topics --zookeeper localhost:2181 --delete
--topic LXF-TOPIC-TEST
```

Find more information about a specific Kafka topic as follows:

```
$ kafka-topics --describe --zookeeper localhost:2181
--topic NEW-LXF-TOPIC
```

The last command comes in very handy when you want to find out how many partitions a topic has. Enter the Zookeeper shell as follows:

```
$ zookeeper-shell localhost:2182
[zk: localhost:2182(CONNECTING) 0]
```

Lastly, you can check the performance of a Kafka topic as follows:

```
$ kafka-producer-perf-test --topic NEW-LXF-TOPIC
--throughput 10000 --record-size 300 --num-records
20000 --producer-props bootstrap.
servers="localhost:9092"
```

The performance of a Kafka topic mainly depends on the number of partitions and the way these are distributed across different servers. Figure 3 *(below)* shows the output of some of the commands presented in this section. Note that the only way to safely backup Kafka is to have replication: there is no specific Kafka utility for backing up and restoring a topic. However, you can backup the Zookeeper State Data that is essential for Kafka to work; again, talking more about this is beyond the scope of this tutorial.

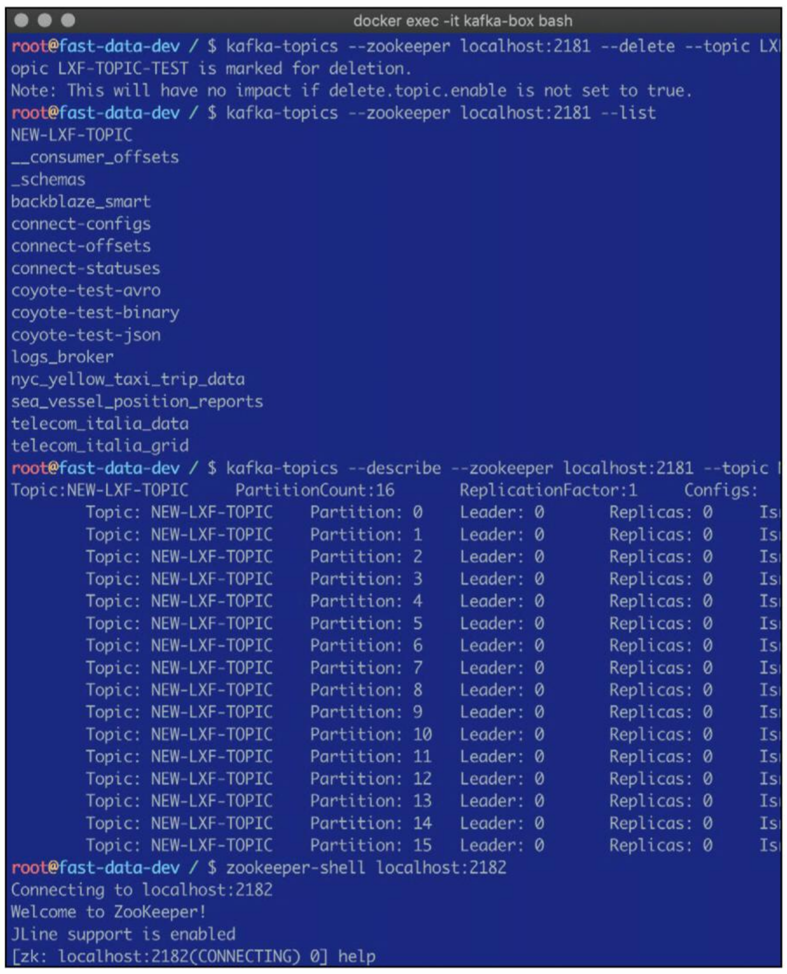


Figure 3: The use of some administrative commands that enable you to delete an existing Kafka topic, print information about a topic and connect to the Zookeeper shell.

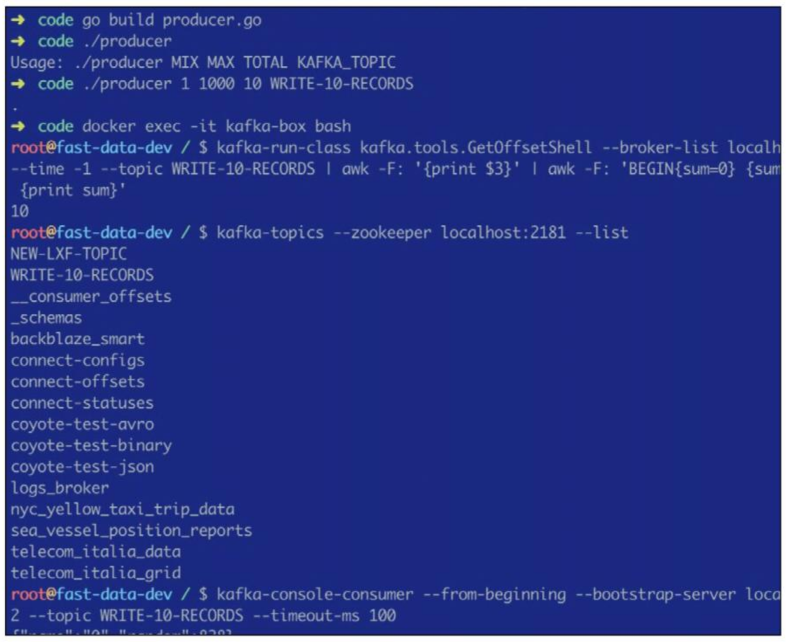


Figure 4: The use of the producer.go utility as well as the results of its execution. As we a using a Docker image, all changes will be lost when we stop the Docker image from running.

Creating a producer in Go

Now we'll learn how to develop a Kafka producer written in Go. The Go producer will store data in Kafka using JSON records. As the Docker image makes some of its services and ports available to the rest of the world, you will not need to connect to the *bash* shell of the Docker image for the producer and the consumer Go utilities to work.

The *producer.go* utility writes a user-defined amount of random numbers that reside in a given range to a Kafka topic, which is also given as a parameter to the program. Therefore, you will need to provide the Go program with four parameters. The critical Go code of the *producer.go* utility is the following:

```
conn, err := kafka.DialLeader(context.Background(),
"tcp", "localhost:9092", topic, partition)
conn.WriteMessages(
        kafka.Message{Value: []byte(recordJSON)},
)
```

The first statement defines the Kafka server that we want to connect to, and the second statement is what writes records to the desired Kafka topic. Note that the format of the records is JSON, which means that you will have to use the `json.Marshall()` function in order to convert your data into JSON – this is the only difficult aspect of the *producer.go* utility.

As this point it's a good idea to download the Go package that will be used for connecting to Kafka – without that package neither the Go producer nor the Go consumer will be able to work. This can be done by executing `go get -u github.com/segmentio/kafka-go`. You can find more information about the used Go library at **https://github.com/segmentio/kafka-go**.

After executing *producer.go* with the required parameters, you can go to your Kafka installation and look for the data, which can be done as follows:

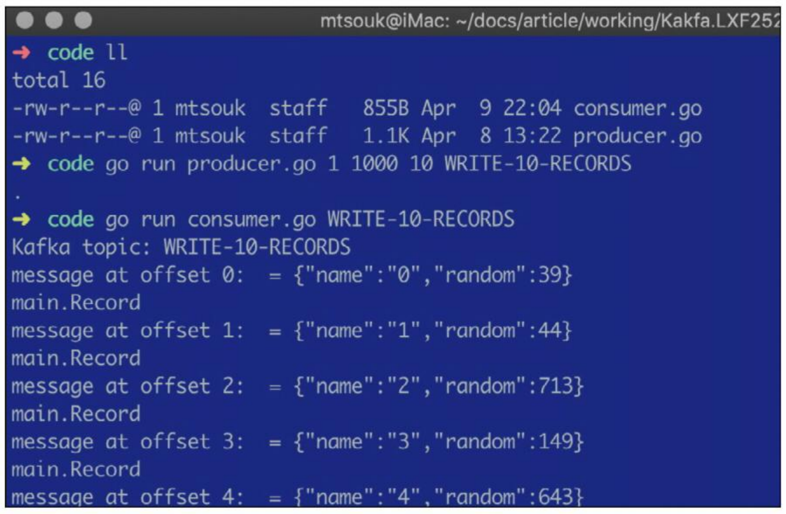


Figure 5: The consumer.go utility in action. As consumer.go does not use Go reflection, it should know in advance the format of the data.

```
$ docker exec -it kafka-box kafka-console-consumer
--zookeeper localhost:2181 --topic <TOPIC_NAME>
```

The output of the *producer.go* program itself is pretty simple, as it prints a dot character on the screen for each 50 records it writes to Kafka. Figure 4 *(see page 90, top right)* shows more about that process, as well as the output of the `kafka-console-consumer` command itself.

Creating a consumer in Go

Let's turn our attention to creating a Kafka consumer written in Go (*consumer.go*) in order to read data from a Kafka topic. The critical Go code of the *consumer.go* utility is the following:

```
r := kafka.NewReader(kafka.ReaderConfig{
        Brokers:   []string{"localhost:9092"},
        Topic:   topic,
        Partition: partition,
        MinBytes:  10e3,
        MaxBytes:  10e6,
})
r.SetOffset(0)
```

The `kafka.ReaderConfig` Go structure holds the details of the Kafka server we want to connect to, including the topic name and the partition number; as we are using a Kafka topic with a single partition, the partition number will be 0. Meanwhile, the `SetOffset()` function call specifies the offset from which the program will start reading data. If you want to read all data from a Kafka topic, use `SetOffset(0)`.

Figure 5 *(above)* shows a part of the output of *consumer.go* as executed from the UNIX shell of the local machine. Because *consumer.go* keeps waiting for data, you will have to end it by pressing Control+C or using *kill*.

Figure 6 *(right)* shows the entire Go code of consumer.go. The simplicity of the code is pretty impressive for such a handy utility. Note that you will need to have the `Record` structure correctly defined for *consumer.go* to work, which means that you should know the format of the JSON data that you expect to get from the Kafka topic in advance. If you want your Go code to dynamically understand the JSON data, you will need to use Go reflection, which is an advanced feature of Go.

Admittedly, as you've seen so far, Kafka is pretty different – but this is not necessarily a bad thing. You will need some time to get used to Kafka and learn how to administer, use and develop applications for it, but you will rewarded by its simplicity and speed.

When you have to store and process huge amounts of streaming messages, Kafka should be the first thing to come to mind – it's great at storing and serving data fast. Additionally, Kafka is great at enabling you to get data from a supported source, filter it, process it and export it to another destination such as another database or service, an FTP site or a plain text file.

On the other hand, when you have to work with static data or small amounts of data, Kafka is still a viable solution that you should consider. LXF

» USEFUL INFORMATION

The name 'Kafka' was inspired by the author Franz Kafka, because – somewhat tenuously – Kafka software is optimised for writing. Kafka is written in Scala and Java, and was originally developed by LinkedIn and donated to the Apache Software Foundation back in 2011. Its design was influenced by transaction logs.

Some of the developers of Kafka created their own company named Confluent, with a focus on Kafka. Streaming data arrives at a very high rate, which means that you need to store it as soon and as quickly as possible. KSQL is a query language for streaming data, developed by Confluent, that enables you to query live data as it comes in. KSQL supports a wide range of streaming operations, including data filtering, transformations, aggregations, joins, windowing, and sessionisation.

Note that windowing enables you to limit live data based on certain criteria in order to perform certain kinds of queries such as aggregations, because live data can never end. You can find more about KSQL at **http://bit.ly/lxf252ksql.html**.

Figure 6: The entire Go code of consumer.go, which is a utility that reads data from a Kafka topic using the JSON format. Note that the program already knows the format of the JSON records it will receive.

PYTHON

Hacking Minecraft with Python

Calvin Robinson creates a custom toolbox for quickly manipulating a Minecraft environment, complete with graphical user interface.

OUR EXPERT

Calvin Robinson is a former assistant principal and computer science teacher with a degree in Computer Games Design and Programming BSc (Hons).

QUICK TIP

Guizero's 'grid' mode is ideal for moving objects around your app based on columns/rows. However, 'auto' mode is also available, in which items are displayed in the order you code them.

Using Python, in this tutorial we're going to hook directly into *Minecraft* to edit our player world and the characters within in it, all using Python. While Python 3 is an incredibly versatile high-level programming language, as you may know *Minecraft* was originally developed using Java, so we're going to have to install a few add-ons to be able to hook directly into *Minecraft* with Python.

Fortunately, *Minecraft*'s developers Mojang has released a version specifically for the Raspberry Pi – the *Minecraft Pi Edition* – that comes with an API for budding programmers, so we're going to take advantage of this. The following tutorial will work whether you're using a Raspberry Pi or a regular desktop variety of Linux.

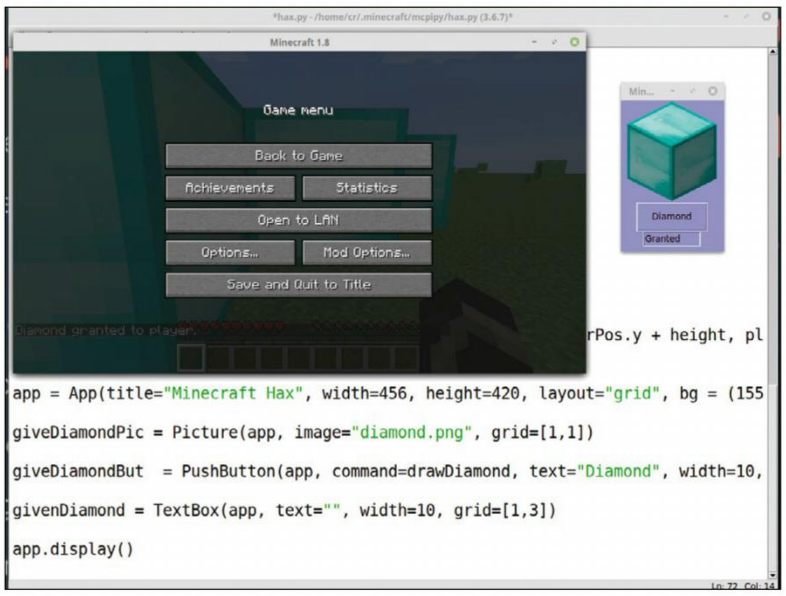
We've put together a package that allows Python to hook into *Minecraft*, called McPiFoMo (**http://rogerthat.co.uk/McPiFoMo.rar**). Simply extract the contents of its **.minecraft** directory into your **~/home/.minecraft** directory and you're good to go.

McPiFoMo includes MCPiPy (from **http://MCPiPy.wordpress.com**) – a *Minecraft Pi* Python plug-in – and Raspberry Jam (**http://alexanderpruss.blogspot.com**), which uses Forge to enable the McPiPy plugin to work on regular Linux *Minecraft*. Provided you have Python installed, which is pretty standard on most distros, no additional software is required other than your favourite text editor or Python IDLE.

Python scripts in this tutorial should always be saved in **~/home/.minecraft/mcpipy/**, regardless of whether you're running *Minecraft Pi Edition* or Linux *Minecraft*. Make sure you run *Minecraft* with the 'Forge 1.8' profile included in McPiFoMo for your scripts to work correctly.

World of GUI

'guizero' is a quick and easy Python 3 library that enables inexperienced programmers to get a GUI up and running. Essentially it taps into tkinter, which can certainly take some getting used to. Download guizero from GitHub (**https://lawsie.github.io/guizero**) and extract the **guizero** directory into **~/home/.minecraft/mcpipy/** and we're ready to start.



Diamonds are most certainly forever. Our first functioning button!

Create a new Python file in **~/home/.minecraft/mcpipy/** and import the two libraries we're going to be using, namely `guizero` and `Minecraft`, and create a new instance of the `Minecraft` function:

```
from guizero import App, PushButton, Slider, Text, Window, TextBox, Picture
from mc import *
mc = Minecraft()
```

Now let's set up the main window of our application:

```
app = App(title="Minecraft Hax", width=456, height=420, layout="grid", bg = (155, 155, 250))
```

Here `app` is obviously the variable we'll be using to refer to this window going forward (the primary window in guizero is always referred to as the app). Use whatever title you like, and set your preferences for width/height. `bg` is the background colour and can use hexadecimal colour codes (prefaced with a `#`) or RGB as above. We're using what's known as a grid layout, which will become clear as we add objects.

As a starting point, let's create a button to gift our player character a diamond block. In guizero we add a new object of the type `PushButton`, specifying the window in which it should appear (in this case `app`), a command for the button (this will be a function we'll get to soon), the text to display on the button (a string – alter accordingly) and the width of the button. The grid location x/y is quite literally the column/row at which the button will appear in that given window.

```
giveDiamond = PushButton(app,
command=drawDiamond, text="Diamond", width=10,
grid=[1,2])
```

In order for our button to do anything, we'll need to create a function further up in our code. Here we're using our first bit of *Minecraft* code:

```
def drawDiamond():
  playerPos = mc.player.getPos()
  mc.setBlock(playerPos.x, playerPos.y + 1,
playerPos.z-1, 57)
  mc.postToChat("Diamond granted to player.")
```

To ensure the diamond block is placed near our player we first find their coordinates, then use `setBlock` to place the block very near to those coordinates, give or take a space. `57` in this case is the code for a diamond block; we could use any blockID here. Google is your friend when it comes to finding these. We've also taken the liberty of posting a message in the game chat to let players know that a diamond block has been placed; this is great for testing purposes as we can see if our button is functioning even if we can't find the block we're placing.

Right at the bottom of our program we'll want to display the actual window for when the program is run:

```
app.display()
```

Enter the Minecraft

In order to test the program we'll first need to open *Minecraft* with the Forge 1.8 profile, start a new game world and then run the Python script. We've found it helps to create a 'superflat' world for testing purposes. If all goes well, when we press the diamond button we should see a diamond appear in front of our player character and a message in chat saying "Diamond granted to player". If you don't see the block you may want to play around with the x/y/z coordinates in the button function.

You can run the script from the Python IDLE, or by typing `python filename.py` directly in *Minecraft*, provided it's saved in the **/mcpipy** directory.

Now that we've got a working button we'll probably want to make things look a little more user-friendly. After all, that's the point in having a graphical user interface. Above and below our `giveDiamond` button (which was in column 1, row 2) we're going to add an icon and a status box. In the following code we've placed a picture in column 1 row 1, and a text box in column 1 row 3, sandwiching our original button:

```
giveDiamondPic = Picture(app, image="diamond.png",
grid=[1,1])
givenDiamond = TextBox(app, text="", width=10,
grid=[1,3])
```

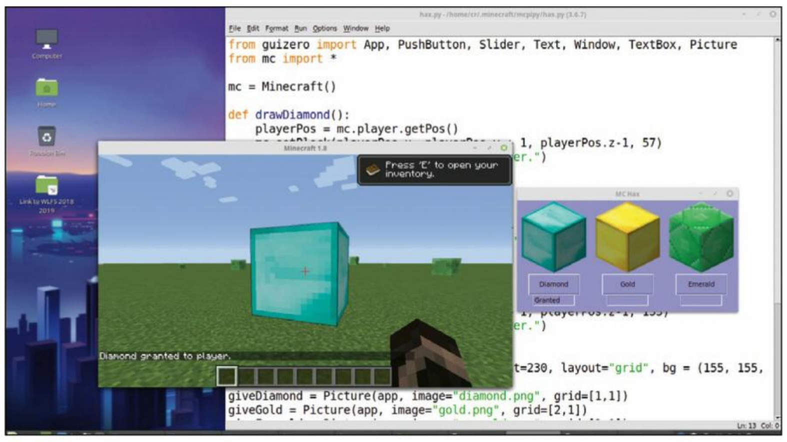
An image named **diamond.png** will need to be saved in the same directory as our **.py** file, of course. You can draw this yourself or find a *Minecraft* diamond image on Google. The text box will become a status box of sorts: we can add the following line of code to our `drawDiamond()` function to change the status of the text box when the button is pressed, thus ensuring we have a visible alert in our app.

```
givenDiamondBox.value = "Granted"
```

Granting more goodies in-game is simply a case of copying the button, picture, text box and function to duplicate this setup to add a row of buttons for granting different items in-game:

```
def drawGold():
  playerPos = mc.player.getPos()
  mc.setBlock(playerPos.x, playerPos.y + 1,
playerPos.z-1, 41)
  mc.postToChat("Gold granted to player.")
  givenGoldBox.value = "Granted"
def drawEmerald():
  playerPos = mc.player.getPos()
  mc.setBlock(playerPos.x, playerPos.y + 1,
playerPos.z-1, 133)
  mc.postToChat("Emerald granted to player.")
  givenEmeraldBox.value = "Granted"
giveGoldPic = Picture(app, image="gold.png", grid=[2,1])
giveEmeraldPic = Picture(app, image="emerald.png",
grid=[3,1])
giveGoldBut = PushButton(app, command=drawGold,
text="Gold", width=10, grid=[2,2])
giveEmeraldBut = PushButton(app,
command=drawEmerald, text="Emerald", width=10,
grid=[3,2])
givenGoldBox = TextBox(app, text="", width=10,
grid=[2,3])
givenEmeraldBox = TextBox(app, text="", width=10,
grid=[3,3])
```

While having fancy buttons to instantly spawn items in-game is great fun, we might also want to occasionally



Diamond, Gold and Emerald buttons placed beneath fancy icons (at 64x64 resolution for uniformity). A diamond has been spawned.

QUICK TIP

Where there's a 'get' there's a 'set'. Any time you're using 'set' for a block or a position, it's often handy to use 'get' first to create a reference point – be it position, rotation, direction, pitch and so on.

» THE LITTLE DETAILS

It's the small things that make all the difference. Try adding a jokey little copyright message to your program to give it an official look:

```
copyrightMsg = Text(app, text="© CR 2019", width=10, grid=[3,9])
copyrightMsg.text_color = "#474747"
```

You'll notice we greyed out the text a little in the code abode. This can be done for any text in guizero, by using a hexadecimal colour code and setting the `text_color` (US spelling) after creating a `Text` object. A good combination of background colours (`bg = #000000`) and fonts can really make your program stand out. The font face/type can also be set along the lines of `copyrightMsg.font = "Verdana"`. Other properties of fonts include: `size` (number), `grid`, `align` (left/centre/right), `visible` (True/False), `enabled` (True/False), and `height` and `width`.

You can get really funky with the `Text` object by calling its methods. `clear()` or `destroy()` can be handy for resetting textboxes (something we might implement on the 'Granted' boxes). `focus()` gives focus to the widget, which is particularly useful when you want a user to be able to write straight into a text box. This would be handy for our admin chat pop-up tool to ensure users can write in the chat box from the moment the window opens.

spawn a custom item that we have no pre-made button for. In the following code we're having to take a few extra steps. By adding another text box we can enable the user to input a specific *Minecraft* itemID/blockID.

In the following function we're storing the value of said textbox in a variable named `itemID` when the 'Spawn' button is pressed. We're then granting that item in-game and printing a message in chat as before. However, we've had to convert the whole message into a string (`giveItemString`) before posting it to chat, because the *Minecraft* API only allows one input.

```
def giveItem():
    playerPos = mc.player.getPos()
    itemID = giveItemBox.value
    mc.setBlock(playerPos.x, playerPos.y + 1,
playerPos.z-1, int(itemID))
    giveItemString = "ItemID %s granted to player." %
itemID
    mc.postToChat(giveItemString)
    giveItemBox.value = "Granted"
giveItemsPic = Picture(app, image="diamondhammer.
png", width=50, height=50, grid=[1,5])
giveItemMsg = Text(app, text="Custom item:", width=10,
grid=[1,6])
giveItemBox = TextBox(app, text="ItemID", width=10,
grid=[1,7])
giveItemButton = PushButton(app, text="Spawn",
command=giveItem, width=10, grid=[1,8])
```

QUICK TIP

Martin O'Hanlon has written a fantastic reference for the Minecraft Python API Library, available at http://bit.ly/lxf252mineapi. It's very useful for finding additional functions built into the MC module.

Automatic for the People

Spawning items in game at the touch of a button can be incredibly time-saving. If you want to build some big rudimentary structures though, you're still going to have to place blocks manually into the world. That is, unless we add an auto-builder to our haxy app…

We're going to need a new window for this one. Define a second window immediately after our original `app` / `App` definition:

```
autoBuilderWindow = Window(app, title="Minecraft
Hax: Auto Builder", width=180, height=160,
layout="grid", bg = (155, 155, 250))
autoBuilderWindow.hide()
```

Notice we set the window to hide by default – we'll create a button and function to 'show' it. In guizero our default app is shown with `display()` , whereas all additional windows use `show()` :

```
def abWindowOpen():
    autoBuilderWindow.show()
openABButton = PushButton(app, text="Auto Builder",
command=abWindowOpen, width=10, grid=[2,7])
```

Now for the actual functionality of the auto-builder. This is the complicated bit. We're essentially taking a value for height, width and depth, then creating a vector from the coordinates with the given blockID. Notice we're using `setBlocks` instead of the singular `setBlock` : this is the command that enables us to give a starting point and end point for our vector build, from x,y,z to x2,y2,z2. We've set the first values to be our player's position, and the second set of values to be taken from the user input.

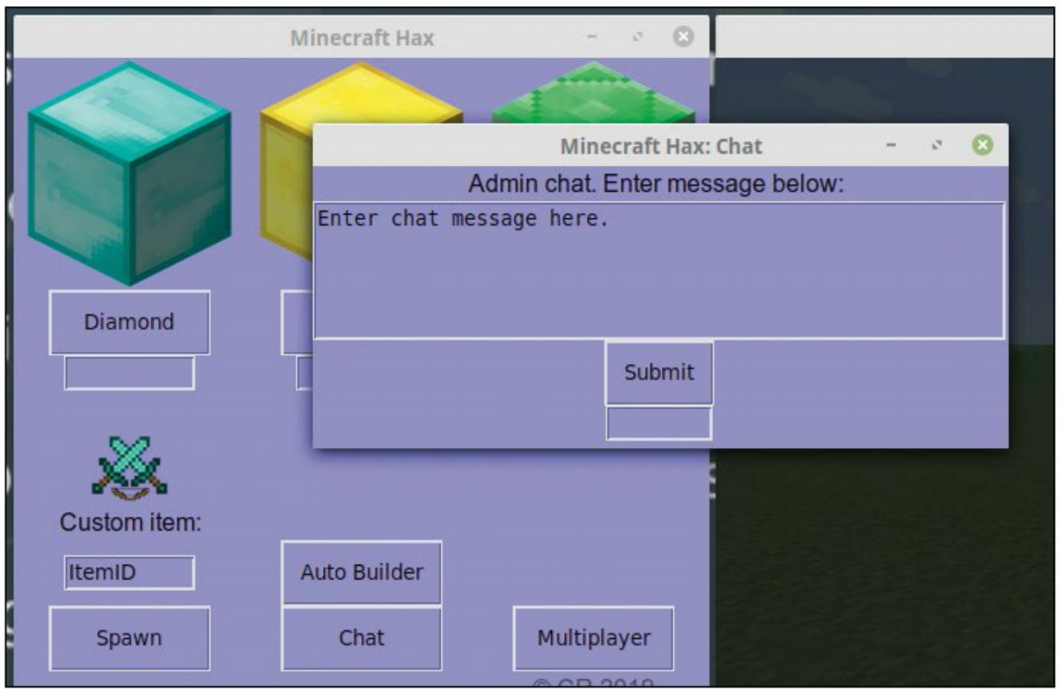
```
def autoBuilder():
    playerPos = mc.player.getPos()
    playerPos.y = playerPos.y + 10
    mc.player.setPos(playerPos.x,playerPos.y,playerPos.z)
    height = float(abHeightTB.value)
    width = float(abWidthTB.value)
    depth = float(abDepthTB.value)
    blockID = float(abBlockTB.value)
    mc.setBlocks(playerPos.x, playerPos.y, playerPos.z,
playerPos.y + height, playerPos.x + width, playerPos.z +
depth, blockID)
    abDone.value = "Built"
abHeightT = Text(autoBuilderWindow, text="Height: ",
grid=[1,1])
abHeightTB = TextBox(autoBuilderWindow, text=" ",
grid=[2,1])
abWidthT = Text(autoBuilderWindow, text="Width: ",
grid=[1,3])
abWidthTB = TextBox(autoBuilderWindow, text=" ",
grid=[2,3])
abDepthT = Text(autoBuilderWindow, text="Depth: ",
grid=[1,4])
abDepthTB = TextBox(autoBuilderWindow, text=" ",
grid=[2,4])
abBlockT = Text(autoBuilderWindow, text="BlocKID: ",
grid=[1,5])
abBlockTB = TextBox(autoBuilderWindow, text=" ",
grid=[2,5])
abButton = PushButton(autoBuilderWindow,
text="Build", command=autoBuilder, grid=[2,7])
abDone = TextBox(autoBuilderWindow, text=" ",
grid=[2,8])
```

Admin tool

Since we've created what amounts to an admin tool, it only makes sense for us to provide an admin chat option. Let's set up another window for this, so as to keep the main app tidy. Again, remember to set the new window as hidden by default, and adding a button and function to open it:

```
chatWindow = Window(app, title="Minecraft Hax:
Chat", width=456, height=185, layout="grid", bg = (155,
155, 250))
chatWindow.hide()
def chatWindowOpen():
    chatWindow.show()
openChatButton = PushButton(app, text="Chat",
command=chatWindowOpen, width=10, grid=[2,8])
```

The code for the chat interface and the button to submit it consists mainly of converting strings into a singular variable to post to chat. We've prefixed our admin messages with "Admin:" but you can alter that:



You may want to replace the place-holder text in the admin chat pop-up window.

```
chatItemMsg = Text(chatWindow, text=”Admin chat. Enter message below: ”, grid=[2,1])
chatItemBox = TextBox(chatWindow, text=”Enter chat message here.”, multiline=True, height=5, width=56, grid=[2,2])
chatItemButton = PushButton(chatWindow, text=”Submit”, command=adminChat, grid=[2,3])
chatItemBox2 = TextBox(chatWindow, text=””, width=8, grid=[2,4])
def adminChat():
  chatMsg = “Admin: “+ chatItemBox.value
  mc.postToChat(chatMsg)
  chatItemBox2.value = “Sent.”
```

A nice next step would be to provide usernames for different admins, which could be done by simply adding another text box for users to place their name into and including the value from that textbox in place of ‘Admin:’. So, something like: `chatMsg = usernameBox.value +chatItemBox.value`.

Multiplayer camera hacking

Next up, the most haxy feature of all: viewing another player character's camera. With the *Minecraft* API we can find player IDs (`mc.getPlayerEntityIDs`) and then set our camera to follow that one instead of our own. Starting with another hidden window and button to open said window:

```
playersWindow = Window(app, title=”Minecraft Hax: Players”, width=456, height=228, layout=”grid”, bg = (155, 155, 250))
playersWindow.hide()
openPlayersButton = PushButton(app, text=”Multiplayer”, command=playersWindowOpen, width=10, grid=[3,8])
def playersWindowOpen():
  playersWindow.show()
  getPlayers()
```

You may notice that we're running the `getPlayers()` function the moment we open the Multiplayer window. This will grab the IDs of all currently connected players, ready to list them in a text box:

```
def getPlayers():
  players = mc.getPlayerEntityIds()
  playersBox.value = players
```

Of course, we're also going to need to provide a method of resetting the camera back to the user's own player character.

```
def setCamera():
  mc.camera.setFollow(players[playerSubBox.value])
def resetCamera():
  mc.camera.setFollow(players[1])
playersBox = TextBox(playersWindow, text=”Players list”, multiline=True, height=5, width=56, grid=[2,1])
playerMsg = Text(playersWindow, text=”Enter a PlayerID to view their camera: ”, grid=[2,2])
playerSubBox = TextBox(playersWindow, text=”PlayerID”, grid=[2,3])
setCamButton = PushButton(playersWindow, text=”Set Camera”, command=setCamera, grid=[2,4])
resetCamButton = PushButton(playersWindow, text=”Reset Camera”, command=resetCamera, grid=[2,5])
```

» GETTING INTO TKINTER

While guizero is fantastic for just getting on with programming a graphical user interface without too much prior knowledge, more advanced users might want to tap into the more complicated settings available in tkinter, and that's entirely possible.

By using the command `add_tk_widget` you can call a tk widget directly from your guizero app:

```
app.add_tk_widget(spinbox)
```

`spinbox` isn't something native to guizero, but is built into tkinter and therefore made available to to our ‘app’. It's worth bearing in mind that any properties will need to be called when initialising a tkinter widget:

```
box = Box(app)
spinbox = Spinbox(box.tk, from_=2, to=8)
box.add_tk_widget(spinbox)
```

Because guizero is a kind of wrapper over tkinter, every guizero widget actually contains a tkinter widget. That means we can also tap into the tkinter properties of guizero widgets. For instance, we've used a lot of PushButtons in our *Minecraft* app; we could alter the highlight colour, a setting guizero doesn't offer, but tkinter does. Using tkinter's config:

```
giveItemButton.tk.config(highlightcolor=”blue”)
```

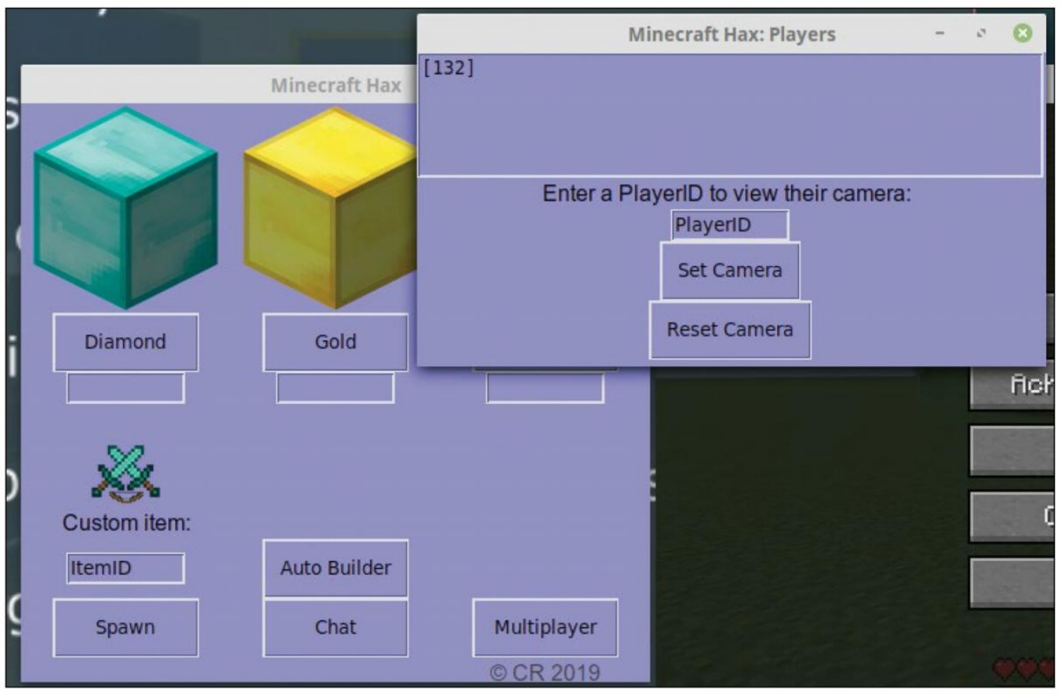
Placing the above code after our definition of `giveItemButton` means that the button now highlights blue. RGB and hexadecimal colour values work here, as well as most basic colour names.

Note that `playersBox` is listening to all the connected players that we discovered when we opened the Multiplayer window. A decent next step would be to provide a way of refreshing that list, perhaps with a simple button that re-runs the `getPlayers()` function.

Now, by typing a player ID from the list into the text box and submitting the button, the camera should change. Quite an effective little hack for spying on other players, this is also often used for ‘spectator mode’ in mini-games.

With this script pretty much complete, we can now spawn items, build structures, take over a player's camera and launch admin chat, all at the press of a button. Feel free to mess around with the layouts and even add new functions! LXF

In this instance we only have one player connected to our game server, their PlayerID being 132.



On the disc

Discover the highlights from this month's packed DVD!

» START HERE

USING THE LXFDVD

Using Linux for the first time can be very confusing. It'll be unlike anything that you've likely operated before, especially if you're used to Microsoft Windows or Apple macOS.

Generally our DVDs are designed to be run directly, which is to say that when you first power on your PC (or Mac) it should 'boot' from the DVD – so before Windows or macOS even starts to load – with Linux running directly from the DVD. This trick is known as a Live Disc. It enables you to try out the various versions of Linux without having to install or change anything on your PC. Just remove the DVD, restart your PC and it'll be exactly as you left it.

While many systems will boot from a DVD when it finds one, many will not. See below for the standard process for enabling booting from a DVD on various desktops and laptop PCs.

The alternative option is to locate the ISO file on the DVD and write this to your own USB thumb drive and attemp to run that. We recommend using *Etcher* from **https://balena.io/etcher** that's available for Windows, macOS and Linux. Good luck!

BOOT THE DISC

Many PCs should boot automatically if they're turned on with a disc in the drive. If not, many offer an early Boot Menu accessed by tapping a key while powering up from cold: F9 (HP), F12 (Dell, Lenovo), F8 (Amibios) or F11 (Award BIOS). Alternatively, use the BIOS/UEFI to adjust the boot order to start with the optical drive. Again, this is accessed by tapping a key during power up, usually Del but sometimes F1 or F2. Some new UEFI PCs require access via Windows: holding Shift select its Restart option. If you're still having problems using the DVD visit: **www.linuxformat.com/dvdsupport**

Mac owners: Hold the C key while powering on your system to boot from the disc.

ON-TREND DISTRO

64-bit

Pop!_OS 19.04

Pop!_OS is the bespoke operating system from Linux system builder System76. Naturally it's been carefully tailored to work on its own hardware (which is great, by the way), but it's based on Ubuntu so it should work on yours too. Awkward punctuation aside, it looks great, with its eye-catching desktop and slick installer. It's more than your average Ubuntu clone too: for one thing it follows a rolling-release model, at least as far as applications are concerned, and for another it enables full-disk encryption by default. It also doesn't include Ubuntu's slightly worrisome data-reporting.

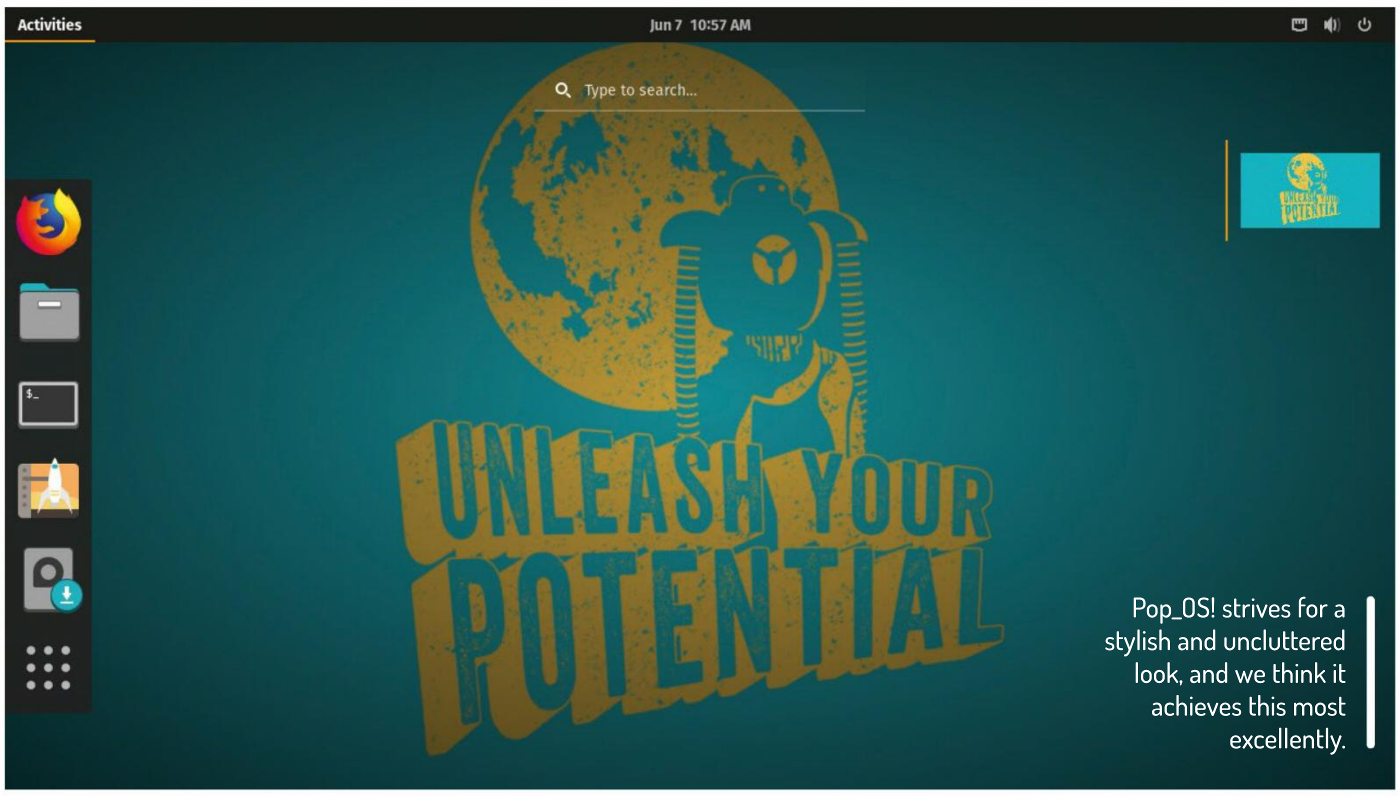
The live Pop edition appears to launch straight into the installer, but don't worry if you just want to try it out. That option appears after you've answered the localisation options. After a couple of questions a 'Try Demo Mode' button will appear in the bottom-left corner. The initial installation is kept deliberately light, but you'll find everything you need in the Pop store.

Pop comes in two versions, Nvidia and Intel/AMD. We've gone for the FOSS Intel/AMD edition. If you have Nvidia hardware and want to use the proprietary driver, you may want to head over to **https://system76.com/pop** and download the Nvidia edition. It's been tweaked to avoid any of the usual blank-screen-on-boot horror stories, and works seamlessly with hybrid (dual Nvidia/Intel) graphics.

You can also add the driver with

```
$ sudo apt install system76-driver-nvidia
```

and all should be well. We have a whole feature about Pop on page 46, so you can find out more about it there.



Pop_OS! strives for a stylish and uncluttered look, and we think it achieves this most excellently.

» IMPORTANT NOTICE!

DEFECTIVE DISCS: For basic help on running the disc or in the unlikely event of your *Linux Format* coverdisc being in any way defective, please visit our support site at **www.linuxformat.com/dvdsupport**. Unfortunately, we're unable to offer advice on using the applications, your hardware or the operating system itself.

NETWORK ATTACHED STORAGE

OpenMediaVault 4.1.22 64-bit

To coincide with our storage feature *(page 32)*, we've included this fantastic distro, which enables you to set up your own killer NAS for storage, backup, streaming, running *Docker* images and, well, anything you want. You can't boot it from the DVD, but write the ISO file (which you'll find in the **OMV** directory) to a USB stick, either with *dd* or with a graphical tool like *Etcher* (**https://balena.io/etcher**) and you can boot that.

Installation is a breeze; we've covered what you'll need to get started in the feature. It has a beautiful web interface that you can access remotely and securely; SSL certificates are easy to set up from said interface. You'll be serving beautiful graphs showing the throughput of your network shares and all kinds of other stats in no time. *OMV* can alert you in real-time via email to any developing situations (like failing or full drives), which should prevent any surprises.

LIGHTWEIGHT LINUX

Peppermint 10 32-bit

Peppermint is a lightweight Linux distribution based on the ever-popular Linux Mint. While the lightweight LXDE desktop, and the fact it's 32-bit, make it ideal for older hardware, it aims to be useful to all sorts. The choice of desktop ensures it runs as fast as hardware allows, and your memory won't be all used before you've even opened a file manager.

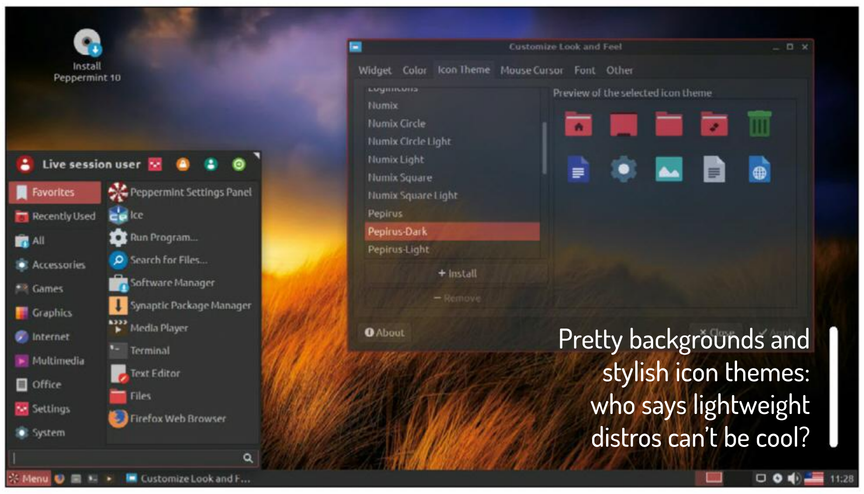
Speaking of such, Peppermint uses Cinnamon's *Nemo*, which is somewhat more fully featured than LXDE's *PCManFM*. For similar reasons it uses Xfce's window manager, menu and panel, keeping to the traditional desktop metaphor, but allowing a few modern luxuries (like type-to-search in the menu).

Billed as a web-centric OS, it enables web apps to be 'installed' to the system menu. Furthermore it runs these in a Site Specific Browser (SSB) instance. The *Ice* tool enables you to configure each SSB, including which browser to use (*Chrome*, *Chromium*, *Vivaldi* and *Firefox* are supported), which icon to use and whether to use an isolated browser.

We know – anyone can make a shortcut to open a web browser at a given URL. But web apps look tidier in an SSB, as there are no crufty menus or toolbars to get in the way. A number of web apps are set up out of the box, including Microsoft Office Online, Google apps and the mighty BetterThanChess.com which uses WebGL to provide a glorious 3D chess experience. It aims to do what Chrome OS has done – provide a stable platform for running web apps – but at the same time still be useful (and in fact excellent) as a traditional OS.

If you're prompted for a username it's 'peppermint', and the password is blank. If this happens it's usually a sign that the desktop isn't going to load, which is usually indicative of a graphics driver problem.

Last issue we didn't have any 32-bit distros so it's nice to include one this time. Peppermint should continue to see updates for as long as Mint does, which is until 2023. LXF



Pretty backgrounds and stylish icon themes: who says lightweight distros can't be cool?

» NEW TO LINUX? START HERE…

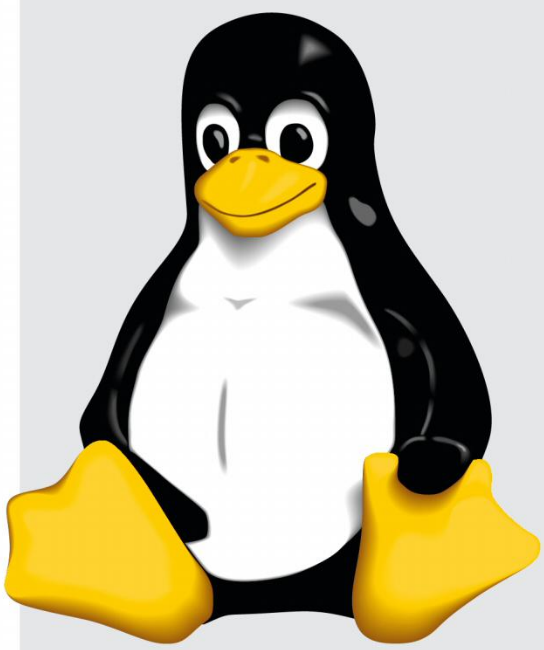
Never used a Linux before? Here are some handy resources:
- Read our quick-install guide http://bit.ly/LXFinstall
- Looking for an answer? https://askubuntu.com
- Want to delve more deeply? https://linuxjourney.com

DOWNLOAD YOUR DVD IMAGES!
Get code and DVD images at: **www.linuxformat.com/archives**

» AND MORE!

READING MATTER

- **Advanced Bash Scripting Guide** Go further with shell scripting.
- **Bash Guide for Beginners** Get to grips with the basics of Bash scripting.
- **Bourne Shell Scripting** First steps in shell scripting.
- **The Cathedral and the Bazaar** Eric S Raymond's classic text explains the advantages of open development.
- **The Debian Book** Essential guide for sysadmins.
- **Dive Into Python** Everything you need to know.
- **Introduction to Linux** A handy guide full of pointers for new Linux users.
- **Linux Dictionary** The A-Z of everything to do with Linux.
- **Linux Kernel in a Nutshell** An introduction to the kernel written by master hacker Greg Kroah-Hartman.
- **The Linux System Administrator's Guide** Take control of your system.
- **Tools Summary** Overview of GNU tools.
- **GNU Emacs Manual** Six hundred pages of essential information!
- **Producing Open Source Software** Everything you need to know.
- **Programming from the Ground Up** Take your first steps.



Protect your online privacy

Stop leaking data all over the internet as you surf: the essential guide to a secure and safe online life.

Better photo management

You don't *have* to use Shotwell – we test the best open source photo managers to see which are best for super snaps.

Music to your ears

Linux sounds awesome for music and we prove it by setting up a digital audio workstation.

Ancestry with Gramps

Track your family tree with the best open source genealogy tools – we get you up to speed on how to research your past.

Monitor everything!

Get to grips with the best monitoring software. We run Nagios on the Pi to help keep a track of everything.

Contents of future issues subject to change – we might be trapped in Jonni's lead-lined bunker

LINUX FORMAT
The #1 open source mag

Future Publishing Limited,
Quay House, The Ambury, Bath, BA1 1UA
Email linuxformat@futurenet.com

EDITORIAL
Editor Neil Mohr
Storage king Jonni Bidwell
Art editor Efrain Hernandez-Mendoza
Production editor Ed Ricketts
Group editor in chief Graham Barlow
Senior art editor Jo Gulliver
Editorial contributors Mats Tage Axelsson, Michael Bedford, Ed Bennett, Neil Bothwick, Christian Cawley, Dan Dart, Tam Hanna, Matthew Hanson, Jon Masters, Nick Peers, Les Pounder, Calvin Robinson, John Schwartzman, Mayank Sharma, Shashank Sharma, Alexander Tolstoy, Mihalis Tsoukalos
Cartoons Shane Collinge
Cover illustration magictorch.com
Raspberry Pi is a trademark of the Raspberry Pi Foundation.
Tux credit: Larry Ewing (lewing@isc.tamu.edu) and The GIMP.

ADVERTISING
Media packs are available on request
Commercial director Clare Dove
clare.dove@futurenet.com
Senior advertising manager Lara Jaggon
lara.jaggon@futurenet.com
Advertising manager Michael Pyatt
michael.pyatt@futurenet.com
Director of agency sales Matt Downs
matt.downs@futurenet.com
Ad director – Technology John Burke
john.burke@futurenet.com
Head of strategic partnerships Clare Jonik
clare.jonik@futurenet.com

INTERNATIONAL LICENSING
Linux Format is available for licensing. Contact the Licensing team to discuss partnership opportunities.
Head of Print Licensing Rachel Shaw
licensing@futurenet.com

SUBSCRIPTIONS & BACK ISSUES
Web www.myfavouritemagazines.co.uk
Email *linuxformat@myfavouritemagazines.co.uk*
UK 0344 848 2852
International +44 (0) 344 848 2852

CIRCULATION
Head of newstrade Tim Mathers

PRODUCTION AND DISTRIBUTION
Head of production UK & US Mark Constance
Production project manager Clare Scott
Advertising production manager Joanne Crosby
Digital editions controller Jason Hudson
Production controller Nola Cokely

THE MANAGEMENT
Chief content officer Aaron Asadi
Editorial director William Gannon
Brand director Andy Clough
Head of art & design Rodney Dive
Commercial finance director Dan Jotcham
Printed by Wyndeham Peterborough, Storey's Bar Road, Peterborough, Cambridgeshire, PE1 5YS
Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU www.marketforce.co.uk
Tel: 0203 787 9001

Future is an award-winning international media group and leading digital business. We reach more than 57 million international consumers a month and create world-class content and advertising solutions for passionate consumers online, on tablet & smartphone and in print.

FUTURE
Connectors. Creators. Experience Makers.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR)
www.futureplc.com

Chief executive **Zillah Byng-Thorne**
Non-executive chairman **Richard Huntingford**
Chief financial officer **Penny Ladkin-Brand**

Tel +44 (0)1225 442244

ipso. For press freedom with responsibility

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier