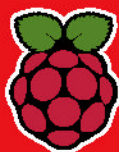




UBUNTU TIPS

Easily batch convert
all your photos



ROBOT CODING

Bring code control to
Raspberry Pi robots

STEAM DECK

How to get more
than games on your deck!



LINUX FORMAT

The **#1** open source mag

POWER UP YOUR MOBILE

with Linux

Revive your
devices, old and new,
with Android and
the Linux desktop!

67
Pages of
tutorials &
features

HOW TO...

- Master GIMP layers and effects
- Get a modern TempleOS reboot
- Manage your installed packages

CODE BOMBERMAN

Relive another classic
game coded yourself

EXTENDING GIT

The ultimate way to
develop your projects

ALGORITHMIC ART

Release the creative
magic of mathematics!



SILVER BLUE

Try the super-secure
future of Fedora



LXF March 2023

FUTURE



PAPER POWER

60% of the energy used to produce paper and paper packaging in Europe comes from renewable sources.

Discover the story of paper
www.lovepaper.org

Source: Confederation of European Paper Industries (CEPI), 2018
CEPI represents 92% of European pulp and paper production





LINUX FORMAT



This issue, we're using open source Android and Linux to revive ageing (and new) mobile devices, so which Android app do you think readers should try out?



Jonni Bidwell

Having just inadvertently wiped a phone, I strongly recommend something like *Neo Backup* for backing up your apps and data. Also, if you like to keep it simple, consider replacing your keyboard, gallery, calendar and such with their counterparts from the Simple apps suite.



Nate Drake

I can only recommend F-Droid. All apps in the F-Droid store contain only FOSS, plus there's a strong emphasis on privacy. The only downside is that you have to install the APK manually – for obvious reasons,

Google doesn't offer it in its Play Store.



Nick Peers

Everyone knows about my unhealthy love for all things media-related, so I was thrilled to discover *Finamp*, a music client for Jellyfin media servers that delivers great audio quality and an excellent set of features, and which costs nothing to use.



Les Pounder

Not an Android app as such, more of a tool to access and control your Android device over USB or Wi-Fi. *Scrcpy* is a multi-platform tool that enables us to see our phone on our desktop, open apps, send messages, and record videos to our PC. Best of all, it's free!



Mayank Sharma

I can't recommend *OsmAnd* enough. It's the best app for accessing OpenStreetMap, which I think is the second best example of a community project after Wikipedia. It's a lot more detailed than its proprietary peers, and if something is amiss, you can plop it in yourself.

Right to repair



When work was insisting on two-factor authentication for work-monitored devices – of course, they insisted on you using your personal device instead of providing one – rather than buying a new Android phone, I used an old HTC One and brought it back to life with LineageOS 17.1, aka Android 10, when HTC dropped support back with Android 5.

It's not exactly what people consider when talking about the

right to repair, but the idea that you can't update or upgrade the software on a device is just as key. Because we've been spoilt with the relative openness of the PC platform for both installing software and upgrading the hardware, it's a shock when you encounter something entirely locked down.

Thankfully, with the Android OSP, the ill-fated CyanogenMod and its better-run replacement LineageOS, many old devices can have a greatly extended lease of life (look at Jonni's MotoG) with up-to-date system software and security patches, which is nice.

With people counting their pennies, extending the life of an existing functional device is an essential money-saving tip. So this issue, we look at keeping those devices around your home ticking a little longer. Now that many people have more phones and tablets than PCs, it's good to know open source can help.

In the rest of the issue, we have the usual mix, from advancing our Pi robot with sensor code to taking a look at how to extend a Steam Deck to more than just games, testing out the immutable Fedora Silver Blue, exploring packaging formats and a big bundle of Linux tutorials. So, no matter if you're on PC or mobile, enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com



Subscribe
& save!

On digital and print:
see page 16

Contents



REVIEWS

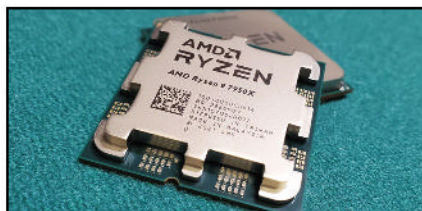
Kingston NV2..... 19

Shane Downing makes an in-depth examination of this budget solid-state drive and concludes that while it's certainly cheap, it definitely isn't cheerful.



AMD Ryzen 7950X..... 20

After much consideration following his look at AMD's new processor, **Dave James** pens the words and takes complete blame for: Zen 4 and the art of CPU maintenance.

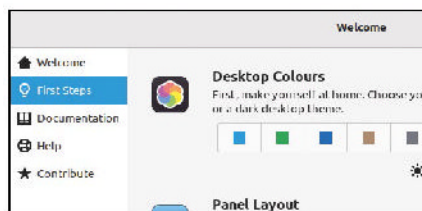


Rocky 9.1..... 21

Nate Drake goes head to head with Red Hat-based distributions this month, so is extremely glad to have the heavyweight Rocky in his corner.

Linux Mint 21.1 Vera..... 22

Nate Drake meets Linux Mint's latest offering, code-named Vera, and comes away bowled over by its stunning looks and ease of use.



Return to Monkey Island..... 24

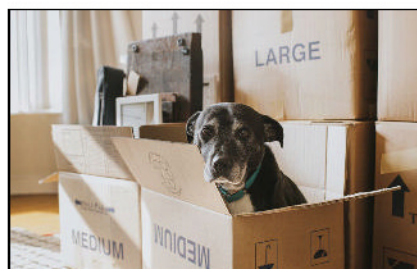
The Management thinks three decades is a perfectly acceptable amount of time for **Will Freeman** to wait to get an under-inflation pay rise.

FREE YOUR PHONE!

Lesser-spotted **Jonni Bidwell** climbs to the top of a 5G mast to extol the benefits of running Linux on your phone. See page 32!



ROUNDUP



Packaging formats..... 26

There's a number of ways of getting software on to your Linux-powered computer, and **Michael Reed** is comparing them to find out which is best.

IN DEPTH



Deck the hardware..... 78

Neil Mohr revisits and tricks out the Valve Steam Deck a year on from its release to demonstrate how you can get even more out of the gaming platform.

PI USER

Pi news 43

Introduced by **Les Pounder**, who's smiling at the new Raspberry Pi Camera Module 3. Plus, the miracle of using just 500 wafers to feed 10 million Pis.

Arduino 2.0.3 44

Les Pounder loves tinkering with hardware, so when Arduino revealed its latest IDE, he had an excuse to tinker even more.

Pi Camera Module v3 45

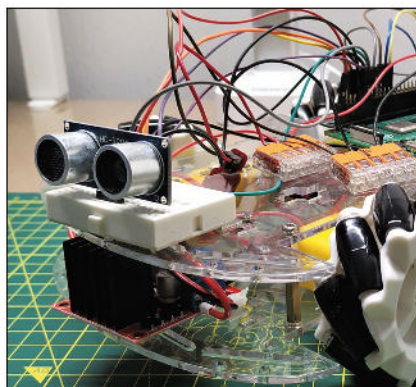
The latest version of Pi's mainstream camera module has autofocus, HDR and wide angle, proclaims **Les Pounder**!

Pico GPIO control with JavaScript 46

Les Pounder strays from Python to show us that JavaScript, the language of the web, is a viable Pico alternative.

Add and monitor control sensors 48

Making sense of the situation, **Les Pounder** shows us how to use sensors with a robot to get about town.



CODING ACADEMY

How to code directly for eBPF 90

Mihalis Tsoukalos explains how to take advantage of eBPF with a range of different programming languages.

Recreate the classic Bomberman 94

When we said "go out with a bang", **Andrew Smith** decided to blow more than just the blinking doors off!



REGULARS AT A GLANCE

News 6

Valve is driving Linux development, Cooler Master's *MasterPlus+* is to get cool open source overhaul, and the sun has finally set on Python 2 for Debian, plus much more.

Kernel watch 10

Answers 11

Sound advice for soundcards, getting Ubiquiti to cooperate, accessing free space, multiple terminal sessions with *Tmux*, sorting out upgrade issues, and more.

Mailserver 14

We take reader criticism on the chin, discover what distro coverage you would like to see, and ponder the possibility of running a tutorial on car diagnostics.

Subscriptions 16

Get your monthly Linux dose and save cash!

Back issues 64

Get hold of previous *Linux Format* editions.

Overseas subscriptions 65

Get *Linux Format* shipped around the globe.

HotPicks 83

Mayank Sharma found solace compiling *HotPicks* as he struggled to fight off a nasty infection his bug magnet brought home from school. This month he looks at *Fotoxx*, *Saber*, *Upscayl*, *LibreWolf*, *Mprocs*, *Junction*, *Myuzi*, *Libre TrainSim*, *Infinite Shooter*, *OTPCClient* and *HydraPaper*.

Next month 98

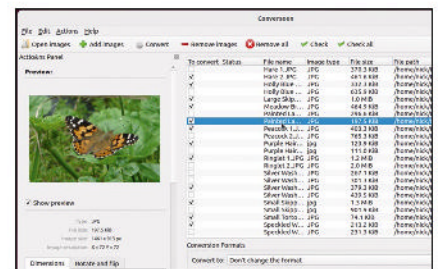
TUTORIALS

TERMINAL: Super system monitor 56

No sniggering at the back as **Shashank Sharma** shows you his *Bottom* and reveals a very useful terminal-based system monitor.

CONVERSEEN: Batch convert images 58

Nick Peers explains how to quickly apply the same changes to multiple images at once with the simple-to-configure *Converseen*.



TEMPLEOS: Open source OS 60

Michael Reed delves into one of the most unusual operating systems that he's ever seen and explores the story behind its creation.

GIMP: Enhanced layer features 66

Karsten Günther takes a look beyond GIMP's standard layer functions. Besides special GIMP operations, tasks provided by G'MIC loom into view.

ALGORITHMIC ART: Get creative! 70

Mike Bedford demonstrates how you can exercise your left and right brain simultaneously by generating stunning algorithmic art.

MAGIT: Make Git easy 74

Aaron Peters uses the *Emacs* Git client Magit to perform a long overdue update of his *LXF* repository, and explains how it performs its magic.

TOP OF THE FOSS!



No changes allowed! 52

Matt Holder explores immutable OSes, containerised apps and Silver Blue to see if they describe the future of Linux distros.

Newsdesk

THIS ISSUE: Valve champions open source » Rival to Google Maps
» Cooler Master API » Debian dumps Python 2 » Google takes a RISC

GAMING

Valve driving Linux open source development

Gaming giant has approached the open source community to enhance graphics support and compatibility of Steam Deck and SteamOS.

Eight years ago, Linus Torvalds said that Valve may save the Linux desktop. It seems his words were prophetic, as Valve's handheld gaming computer, the Steam Deck, comes with its own distro, SteamOS.

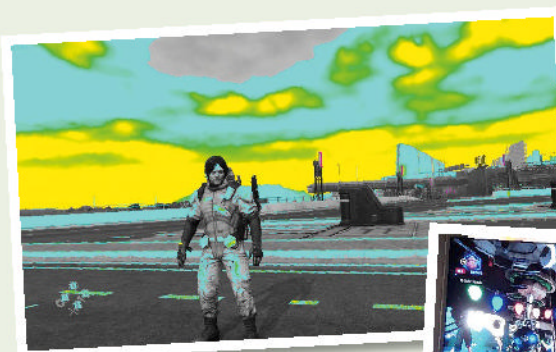
Being Linux-based, Windows games on the Steam Deck have to run through Steam's own Proton compatibility layer. In theory, this saves developers the trouble of creating a Linux port – and, indeed, many titles do work out of the box.

Still, Valve's realised the value of the open source community in helping to keep on top of the patches and updates needed. In an interview with The Verge, Steam Deck designer Pierre-Loup Griffais revealed that Valve has hired over 100 open source developers to work on the Proton compatibility layer, as well as the open source Mesa graphics driver and the Vulkan API.

Griffais added, "There's a larger strategy effort to coordinate all these products and set up an overall architecture, to make sure they can help PC gaming technology thrive in a Linux-based environment."

One of the programmers hired is Joshua Ashton, who's also one of the developers of HDR (high dynamic range) video for Linux. His work on Valve's Gamescope project paid off in January when Griffais announced on Twitter that HDR can now be enabled in real games, posting pictures of *Halo Infinite* and *Death Stranding* as played on his Linux desktop.

Ashton also posted a picture of himself playing *Death Stranding* along with a heat map to demonstrate the benefits of HDR. This teases



Dev Joshua Ashton shows off a heat map of *Death Stranding* using HDR: "Anything not greyscale is brighter than 100 nits." Credit: *Death Stranding*, 505 Games

the prospect that we might see a new Steam Deck 2 that swaps out its LCD screen for an OLED, as with Nintendo's Switch.

It's easy to be cynical about Valve's considerable contribution to open source projects, such as *Wine* (on which *Proton* is based), various graphics drivers, the KDE Plasma desktop (which the Steam Deck uses) and proposing changes to the Linux kernel to make it more gaming friendly. After all, it's in Valve's interests if its Linux-based gaming computer supports as many titles as possible.

Still, Valve does seem to be committed to open source in broad terms, too. SteamOS v3.4 has disabled support for patented video codecs such as H.264 and H.265. Reactions have been mixed, because it can prevent video hardware acceleration using a device's GPU. Skilled coders can re-enable the codecs or install an open source alternative, such as OpenH264, but Valve has clearly drawn a line in the sand.



Pierre-Loup Griffais tweets a picture of a Deck using HDR with an LG OLED TV, but admits there's a "ton of work" to do. Credit: Deep Rock Galactic, Coffee Stain Publishing

MAP APPS

Killing Google Maps

Four tech giants ally with the Linux Foundation to source open map data.

Google Maps comes installed on almost all Android phones, so the tech giant's app dominates the navigation scene. In 2022 alone, it was downloaded over 13 million times. It's unsurprising, then, that a number of tech firms have banded together against the Google Goliath. In December 2022, the non-profit Linux Foundation announced the creation of the Overture Maps Foundation.

The foundation will be an open source programme for curating and collating map data across the world. TomTom, which has struggled against Google's market dominance, is one of the key players. Other backers include Meta, Microsoft and Amazon – so no small fry.

You may note that those other three don't make a point of selling navigation software, but Facebook (as it was) did buy Mapillary in 2020, which specialises in programs to build street maps. AWS also offers Amazon Location Service to allow developers to incorporate location data in their apps.

The foundation's website says at first it will release "basic layers including building, road and administrative information" starting in early



The Overture Maps Foundation will start releasing data early this year, with the backing of four major tech firms.

2023. As Google Maps includes this data with Street View, there is quite a mountain to climb.

The aim of the project seems similar to that of OpenStreetMap, which has been around since 2004 to encourage the free sharing of geospatial data, formed from existing data sets and contributions by volunteers. The Overture Maps Foundation addresses this in its FAQ, promising to make its data available to OSM.

Whether any of this will have any effect on Google's near monopoly on mapping software remains to be seen.

OPINION

THE ART OF AI



Donnie Berkholz is senior vice president, Product, at Percona.

“ Everywhere you look, people are discussing the AI chatbot ChatGPT. It's been applied to write email campaigns, code and articles. It's been banned in some schools as students use it to write essays. Singer Nick Cave called ChatGPT's output in the style of one of his songs “a grotesque mockery” and “a travesty”, so safe to say he is not a fan.

AI based on deep learning takes huge sets of training data, creates models based on that data, then applies this model to real-life scenarios. Using this data to predict what might happen next, AI can recommend options for decisions based on what it has previously seen.

AI can also be used for areas like database management. We can encode years of expertise around databases and share it with the community to resolve performance and security issues automatically. By creating training data based on what ‘good’ looks like, we can apply that experience faster and more widely, such as spotting when security best practices have not been followed.

AI relies on the knowledge and experience of what has happened before. This comes from people knowing what ‘good’ looks like. By understanding what this tech is suitable for, AI can help improve database management for all.

HARDWARE API

Cooler Master to go open source?

MasterPlus+ to get cool open source overhaul.

Cooler Master offers a range of air and liquid CPU coolers, so is very popular with gamers. To date, though, in order to fine-tune cooling options, you had use CoolerMaster's proprietary Windows-only MasterPlus+ software suite.

But there is hope. At CES 2023, Cooler Master announced it would redesign the suite to be open source, as well as include an API.

Still, a fully-featured Linux version of MasterPlus+ won't be available immediately. When pressed by CES attendee Albert of www.boringtextreviews.com, Cooler Master



MasterPlus+ will get an open source overhaul, along with an API for supporting third-party plugins.

admitted that by switching to open source, the suite may work with other coolers besides its own. However, currently this would mean you won't be able to use CM-specific features such as TMS (Thermal Management System). Cooler Master has also announced support for running software in the cloud. Time will tell if the new suite and API will be as cool as it claims.

One very cool feature to incorporate via the new API would be to integrate OpenRGB with the Cooler Master lighting controller. This would allow the suite not only to support the light show on Cooler Master products but also other RAM chips, like the Corsair Vengeance Pro series.

OPINION

FOLLOW ME!



Keith Edmonds is MD of Tiger Computing, which provides support for businesses using Linux.

“For many, open source software equates to “no charge”. That’s usually true, but recent events at Twitter have highlighted another, more important, advantage.

In case you missed it, Elon Musk bought Twitter and made some changes that upset some people. The result was a number of Twits exploring alternatives.

I’ve never been a big Twitter user, but I recently decided I wanted to move away from Facebook. Like the Twitter migrants, I found Mastodon.

Mastodon is closer to Twitter than Facebook, but it isn’t either of them. It’s open source and it’s federated, meaning anyone can run a Mastodon server and they all talk to each other. Think email: your email server and mine may be run by different people but they can communicate.

There’s no algorithm pushing posts into your feed on Mastodon. You follow the people and hashtags you want to. You are the algorithm. No adverts. And federated means no one can take over Mastodon.

It takes a little more effort to get started than Twitter or Facebook do, but the result is freedom: the ultimate advantage of open source software. Me? I’m @backseat@mastodon.org.uk.



DISTRO

Debian kicks out Python 2

The sun has finally set on Python 2 for Debian.

While Debian is praised for its stability, users know this comes at the price of using older programs. While Python 2 was officially retired in January 2020, for the sake of continuity, Debian 11 (Bullseye) came with it bundled.

Debian has now virtually removed Python 2 altogether from the ‘testing’ and ‘unstable’ releases. The **python** package is now gone, as is **/usr/bin/python**. Users of legacy Python 2 apps can still manually install the **python-is-python2** package to get them running.



Python 2 is being unpowered by being removed from the operating system.

This change does only apply to the ‘testing’ and ‘unstable’ versions of Debian. The ‘stable’ branch of Bullseye still has it but developers are working on removing it from Debian 12 (Bookworm) altogether.

The Python project also has a detailed web page on how to use *Lib2to3* and Python tools such as *Modernize* to convert existing scripts.

Debian 11 users needn’t rush to their interpreters right away, though, because the stable LTS version of Bullseye will still be supported until the end of June 2024.

ANDROID

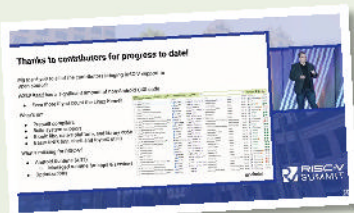
Google to add RISC-V support

Android could turn to open source ARM alternative.

Last month, at the RISC-V summit, Android’s director of engineering, Lars Bergstrom, voiced his support for supporting RISC-V ISA (Instruction Set Architecture) on Android.

Currently, the ARM architecture services almost all modern phones but this can be problematic, given in 2022 the company filed a lawsuit against Qualcomm. RISC-V, on the other hand, is open source. This means any company could use RISC-V compatible software and chips cost-free.

The biggest obstacle is implementing Android RunTime (ART) support to allow developers to emulate apps. Google hopes to do this later this year.



Android’s Lars Bergstrom, presents his road map for support for RISC-V.

VIDEO

Vulkan Video

Khronos finalises Vulkan video extensions.

In April 2021, Vulkan released a set of provisional extensions named Vulkan Video for integrating hardware-accelerated video compression/decompression into its API.

In December, Vulkan’s developer, the Khronos Group, released an upgraded version of these extensions that it claims “expose core and decode Vulkan Video functionality to provide fully accelerated H.264 and H.265 decode”. There are also plans to support the A1 and VPN9 video extensions.

All modern GPUs contain their very own engines for video decoding, encoding and acceleration. Vulkan Video adds video decode and encode queues to support this. It includes both universal core extensions that support a variety of codecs, as well as extensions designed specifically for certain types.

Given the number of APIs already in the wild for video hardware acceleration, such as Vaapi, Quick Sync and Vdpau, it remains to be seen if Vulkan can fulfil its ambition to establish a universal standard.

Still, early adopters include both the mighty Gstreamer and Ffmpeg frameworks, so Vulkan Video might well live long and prosper.

Distro watch

What's behind the free software sofa?

ALPINE 3.17.1

A "minimalist OS designed primarily for routers, servers, firewall hardware and VPNs", Alpine boasts a small download fingerprint, can be set up quickly and even runs in RAM. It uses *BusyBox* to share app resources and has a hardened kernel. The latest release includes a fix for a rather worrying OpenSSL bug (CVE-2022-3996), as well as security upgrades to *Chromium*, *VLC* and *Ruby*. Find out more at www.alpinelinux.org/posts/Alpine-3.17.1-released.html.



Lightweight as mountain air, yet solid as a rock, it's Alpine.

PISI LINUX 2.3.1

Based on Pardus OS, originally developed with support from the Turkish government, Pisi, like Pardus, uses the *PISI* (Packages Installed Successfully as Intended) Python package management system and focuses on desktop users. Setup is easy thanks to the walkthrough from the built-in *Captain* manager. The latest release, Pomegranate, includes the *LibreOffice* suite once again, as well as an updated kernel and *Telegram* desktop. Read more at <https://pisilinux.org/blog/content/123-04-01-2023-updates.html>.



Pisi is user-friendly and comes with certain media codecs and internet apps pre-installed.

ROBOLINUX 12.09

This rather quirky OS is based on the latest LTS version of Ubuntu (in this case, 22.04). There are both free versions and paid ones with extra privacy options. Robolinux also supports running Windows XP and Windows compatible apps out of the box via virtual machines. The latest release uses kernel 5.15 with greater support for Wi-Fi and next-gen CPUs. It comes with a minimal set of apps including *LibreWolf*, *Deluge*, *GIMP* and *VLC*. You can read more at <https://robolinux.org/download.html>.



If you can't leave Windows behind altogether, Robolinux may be the OS for you.

NOBARA 37

The Nobara Project describes its OS as a "modified version of Fedora with user-friendly fixes". In particular, it has tried to make Nobara more friendly for gamers, bundling items such as *Wine* dependencies, *OBS Studio*, third-party codecs like *Gstreamer*, support for Nvidia drivers and much more. Nobara, which was first released in 2022, also prompts you to install additional proprietary codecs post-install. Nobara 37 incorporates kernel 6.0.16 with various patches to support HDR, the Steam Deck and Mesa. Find out more at <https://nobaraproject.org/download-nobara>.



Nobara contains a bundle of pre-installed programs and patches to make Fedora gamer-friendly.

OPINION

WINE ON WAYLAND



Alexandros Frantzis is a senior software engineer at Collabora.

I want to share the exciting progress we made in 2022. The focus was on maturing the Wayland driver and keeping up to date with the *Wine* upstream internal changes. This involved, among other things, splitting the driver into a PE and Unix part, updating it for the latest internal driver APIs, and making preparations to support *WoW64*.

A significant improvement is support for cross-process rendering, required by *Chromium/CEF* apps. Last year, the driver could run *Chrome* with the `--in-process` command-line option. *Chrome* is now supported without any special flags, and is fully GPU accelerated on both *OpenGL* and *Vulkan*.

This update also brings enhanced support for the *linux-dmabuf v4* Wayland protocol (aka *dmabuf-feedback*), which allows compositors to dynamically send information about optimal formats and modifiers, for example, depending on the surface presentation mode (full-screen versus windowed).

We also made a lot of smaller fixes and enhancements that in aggregate have greatly improved driver stability. These were a result of both testing we performed and reports from people trying out the Wayland driver, for which we are very grateful.

OPINION

HI, 2023!



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

“I believe 2023 will be another year of transitions. Hopefully, the pandemic situation will continue to improve, but the fear of further global disruption remains. While I hope we don't see a giant recession, everyone is being asked to do a bit more with fewer resources, and the squeeze is likely to continue.

This manifested a little last year in the consolidation of various small open source outfits into larger companies. I suspect this will continue as everyone begins to plan for where to be when the music stops (if it stops). On a related note, I also would not be surprised if we hear more about the prospect of one or more significant retirements from the kernel and broader open source community (we're not all spring chickens).

One thing I do hope for in 2023 is that we continue to make our communities more accessible and approachable. Many of us began using Linux in a time that was defined by the use of email, IRC and Emacs. But that isn't the way of the modern world. The next generation is more likely to use Discord than IRC, and almost anything other than email. A good new year's resolution would be to embrace more of the tools of the future in order to meet the next generation of developers where they actually live.

Kernel Watch

Jon Masters keeps an eye out for all the latest happenings in the Linux kernel, so you don't have to.

Linus Torvalds gave everyone (whether they celebrate the festive event or not) a timely Christmas present with the release of Linux 6.2-rc1 (Release Candidate 1). This followed a two-week merge window during which disruptive changes were allowed to be added to the new kernel, and we are now well into the stabilisation period that normally follows the first RC release. Typically, there are seven or eight Release Candidates prior to the final release, and then the whole process repeats over again.

As Linus himself said, “6.2 looks like it's a bigger release (certainly bigger than 6.1 was).” One of the big new features continues to be that of enabling support for the Rust programming language. Linux 6.1 added some initial hooks that technically enabled you to compile a Hello World kernel module, but it wasn't possible to do much beyond that in 6.1. And neither is it possible to do much more in 6.2, but new capabilities are being added – slowly.

One of the biggest challenges that currently needs to be overcome is the fact that Rust is a memory-safe language, meaning that it is designed to make it impossible to have a buffer overflow or to deal with a NULL pointer dereference. The problem is that software projects such as

Linux are full of weird funky memory dereferences, structures filled with pointers, and so on. And this means that a lot of adaptation is required through special interfacing code.

Internally, 6.2 adds support for naming anonymous (for example, normal) memory areas that are shared between processes, and places an upper limit on the number of times the kernel can “oops” before it gives up and subsequently reboots. This latter feature deserves diving into a little bit more. An “oops” refers to a situation in which the kernel attempts to perform an illegal memory access (the kind that Rust definitely outlaws but happens all the time in existing code) that would ordinarily result in a program crashing (for example with an error known as a segmentation fault).

When this happens inside the kernel, it instead prints out an error backtrace (log of how this happened), kills whatever task is currently running (without cleaning it up), and tries to continue. When the current task (process) is killed, it may be using resources that are not freed correctly, and consequently it may leak those resources for ever. A skilled attacker could exploit such behaviour through back-to-back oopses, and so a limit can now be placed on how many will be allowed before we simply crash.

» ONGOING DEVELOPMENT

A lengthy thread was started last autumn titled “Reg the next LTS kernel (6.1?)” in which Kaiwan N Billimoria, who is working on a second edition of a *Linux Kernel Programming* book, wondered if Linux 6.1 would become the next Long Term Stable kernel supported by Greg K-H (Kroah-Hartman) for several years of backports. Typically, such kernels are used by Linux distros for their longer-term releases, and an LTS is useful for a book to target.

Traditionally, Greg makes the call on the LTS at the end of the year. This time, he retorted, “You tell me please,” asking the community to say if it was committing to test 6.1 and help him to make it a successful

LTS. A number of folks replied, but as of yet there is no new committed LTS release.

Juri Lelli posted a Call For Papers (CFP) announcing: “After a couple of years' break, OSPM [Power Management in the Linux Kernel] is back!” The organisers want “discussions on the broad scope of scheduler techniques for reducing energy consumption while meeting performance and latency requirements, real-time systems, real-time and non-real-time scheduling, tooling, debugging and tracing”. OSPM is not the only event to return this year. Josef Bacik announced that the Linux Storage, Filesystem, Memory Management and BPF Summit will also be back.

Answers



Neil Bothwick
has 99
problems but
Linux isn't
one of them!

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

Q Mac Format

I have knocked 35GB off the SSD in my 2017 MacBook Pro: 32GB for Ubuntu Mate 22.04 LTS and 3GB for the boot partition. Except for the soundcard, Ubuntu Mate was working right out of the box. I am pretty sure there is Intel High Definition Audio onboard, but being a Mac user since Snow Leopard, I am a little rusty in finding and installing drivers. Can you please tell me how to turn on sound?

My data is currently on a 150GB HFS+ partition. I would like it to be equally available (and manageable) on Mac OS Sierra and Ubuntu Mate. What filesystem do you recommend?

Niels A

A

Before you do anything else, go into the mixer and make sure the soundcard is not muted; some setups do this as a default. If that doesn't help, the next step is to find out which sound device you have. Open a terminal and run:

```
$ lspci
```

You may need to run this command with `sudo`. You'll see a list of devices, each with a number – find your sound device, for example:

```
00:1f.3 Multimedia audio controller: Intel Corporation...
```

Use the `-s` option to limit the output to a specific device and `-k` to show the kernel module it uses – device drivers are implemented as kernel modules in Linux. So, for the above example you would run:

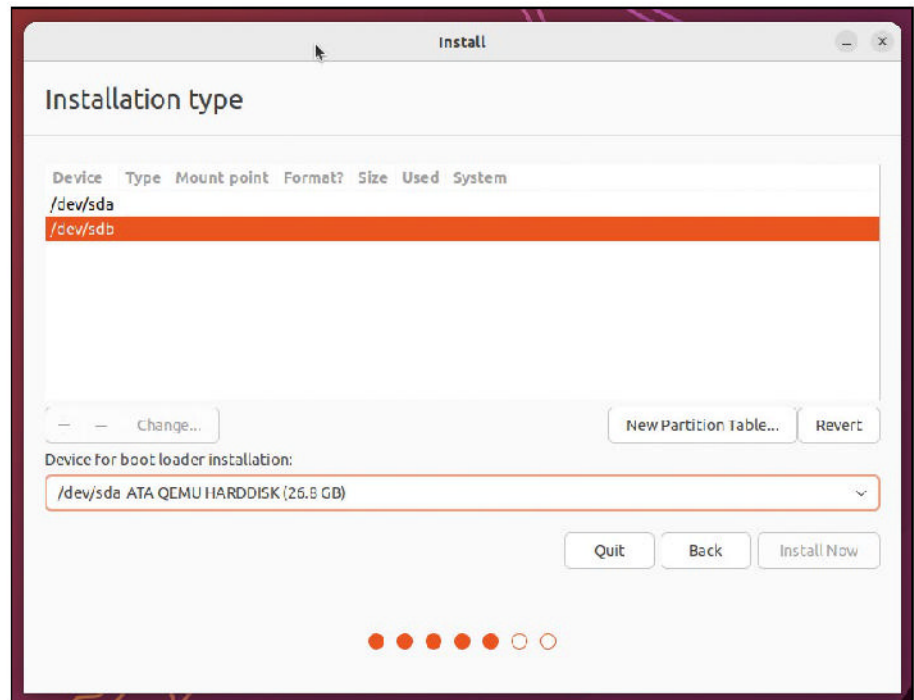
```
$ lspci -k -s 00:1f.3
```

If a module is found, the card should work, but you may have to select a different profile. If it does not have a module loaded, a quick trip to your favourite search engine should tell you which module your card needs. You can test it by forcing it to load in a terminal:

```
$ sudo modprobe -v name_of_module
```

If that works, you can force the module to be loaded at boot by putting its name in a file in `/etc/modprobe.d`. The name of the file is unimportant but it must end in `.conf`.

If the correct module is loaded and you still have no sound, you may be running the wrong profile and trying to send sound



Ubiquity enables you to change the destination disk of both installation and bootloader, if you dive into the alternative options.

to a non-existent device, such as an HDMI port. Run *Pavucontrol* to see and select from a list of available sound profiles.

There's a number of command-line tools for investigating the system, but there are also all-in-one tools such as *Inxi* (<https://github.com/smxi/inxi>) and graphical tools such as *Hardinfo* (<https://github.com/lpereira/hardinfo>).

For the HFS+ data partition, you can mount it in Linux but you have to disable journalling for that filesystem in Mac OS or Linux can only mount it read-only. While you are booted into Mac OS, open a terminal and, assuming your partition is at `/data`, run:

```
$ sudo /usr/sbin/diskutil disableJournal /data
```

It is inadvisable to disable journalling for the system partition. Now you can instruct Linux to mount it at boot by adding this line to `/etc/fstab`:

```
/dev/nvme0n1p4 /data hfsplus defaults 0 0
```

Once again, `/data` is the mount point (it must already exist) and `/dev/nvme0n1p4`

is the shared partition, according to the screenshot you sent.

There is a page on using MacBooks on the Debian wiki, most of which also applies to Ubuntu: <https://wiki.debian.org/MacBook>.

Q Uncooperative Ubiquity

I like to run several Linux systems on various hard drives, and this is usually easy with most Linux systems. But my knowledge of computers is not good. I do think things were easier in the old BIOS days. Now I have to use UEFI, and it is not always possible to switch to legacy mode in the settings.

Most installers are happy to install to the hard drive you select. You plug in your hard drive, and it appears on the boot menu (accessed by pressing F7 or some other F key). The easiest is Calamares, which seems to do exactly what you tell it to. But Anaconda (Fedora) and OpenSUSE are fairly obedient, too. Unfortunately, Ubiquity (used by Ubuntu

and Mint) seems to insist on having a presence on the UEFI of the first hard drive, and then every installation calls itself **ubuntu**. (no I'm Ubuntu!-ED) One exception is Lubuntu, which uses Calamares, and is easier to grapple with.

Is there a way of forcing Ubiquity to restrict itself to a selected hard drive, like the other installers?

Jorgen Stepputat

A Ubiquity lets you install where you want, but you have to know where to look in the options. When you get to the screen titled Installation Type, select Something Else. You now need to set up the partitioning manually. If you are using a separate disk for each distro, you only need EFI, root and swap partitions. Below the partitioning section is a drop-down to select the Device For Bootloader Installation. Choose your disk here and Ubiquity should install everything on the same hard drive. However, that is not the end of the story as EFI firmware expects to find the ESP (EFI System Partition) on the first FAT formatted partition on the first hard drive. While you can often select a different boot drive, effectively making that drive the first, not all firmwares handle this equally well.

Another option is to install the operating system to a separate drive, but keep all the bootloaders on the first drive. You then run into the issue you have already discovered, that they are all named **ubuntu**, but you can get round this. There is no direct method of renaming an entry using the standard *Efibootmgr* tool; you need to delete an entry then recreate it with the new name. Fortunately, there is a script to do this for you, available from <https://github.com/s-n-ushakov/rename-efi-entry>. Once you have downloaded the script and made it executable, you can rename an entry with:

```
$ sudo ./rename-efi-entry oldname
newname bootnum
```

If you want to use spaces in either of the names, remember to enclose them in quotes. The **bootnum** argument is needed if there is more than one entry with the same name – such as your umpteen **ubuntu** entries. Run *Efibootmgr* with no arguments to list your bootloaders, and use the numeric portion of the BootNNNN designation for the loader you wish to rename.

Q Ubuntu two-step

Why am I seeing a 'filesystem is full' error, when there appears to be free

space? Running **df** in a terminal shows there is free space but I cannot write to the disk. Is this a corrupted filesystem, a failing disk or something else?

Oliver Iqbal

A Is this a Linux partition using the ext4 (or earlier) filesystem? The **df** command tells you the filesystem in use if you add the **-T** option, or you can use *Findmnt* to list all mounted filesystems and their details. Assuming it is an ext filesystem, there are two reasons it can report the filesystem full when there is space. The first is that the ext filesystems reserve a certain amount of space for the root user: 5% by default. This is so that a user cannot render the system unusable by filling up the root partition. You can see if this is the case by trying to write to the filesystem as the root user, for example:

```
$ echo "test" | sudo tee -a /test
```

If this is the case, you can make a little more space available to your user by reducing the reserved space. The 5% default made sense back when drive sizes were measured in hundreds of megabytes, or even small numbers of gigabytes, but 5% of a 2TB drive is 50GB! The *Tune2fs* program can be used to alter the reserved amount, even with the

» A QUICK REFERENCE TO... TMUX

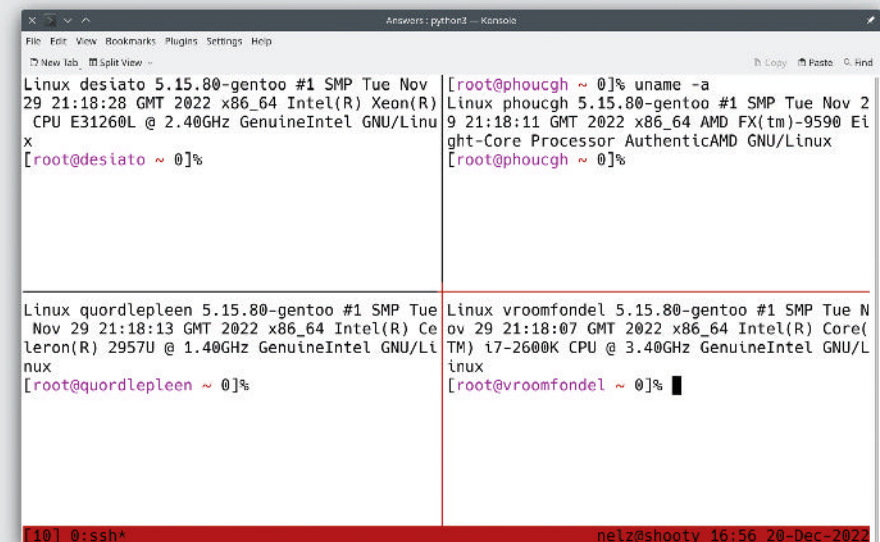
Once you become comfortable with the Linux command line, you find yourself using terminals more and more, often several at once. A tabbed *xterm* is one way of keeping track of multiple terminal sessions, but this is not much use when working with a remote login or virtual console. That's where *Tmux* comes in, enabling you to run more than one session in the same terminal, but that's just the start.

Run *Tmux* in a terminal and not much appears to happen except a status bar appears at the bottom of the terminal window, but you are now in another terminal. Start a long-running command, such as `$ watch date`, then press **Ctrl+b c** and it's gone. Run another command then press **Ctrl+b n** and you are back with the program, still running. **Ctrl+b c** created a new terminal, leaving the other one intact, **Ctrl+b n** switches between them.

Now press **Ctrl+b d** to exit *Tmux* and return to your original shell. Open another terminal and run:

```
$ tmux attach
```

You are now back in your original *Tmux* session. *Tmux* also works over



Using *Tmux* to run the same command on four machines over SSH simultaneously.

SSH. You could SSH into your home computer from work, for example, open a *Tmux* session and start running something, then exit.

Later, you can reconnect to your session, either from work or when you get back home, and pick up where you left off. Even if the SSH connection is

terminated, the programs running in *Tmux* will continue.

There are plenty more options, all detailed in the man page. You can split the current display, more than once if you want, running different programs or even connecting to different computers in each pane.

filesystem mounted, for example:

```
$ sudo tune2fs -m 2 /dev/sda2
```

Do not be tempted to set this to zero on the root filesystem, although it is fine for a partition that is used only for user data, such as a separate **/home** partition.

The other possibility is that you have run out of inodes. These are a part of the filesystem structure and the number available is fixed when the filesystem is created. It uses one inode for each file, so a filesystem with a large number of small files can run out of inodes before it uses up all the space. You can see your inode usage with **df** by adding the **-i** option. The bad news is that it's not possible to alter the number of inodes on an existing filesystem; the only way to gain more is to copy off all your data, reformat with **Mke2fs** and then copy the data back. There are several options for **Mke2fs** to control the number of inodes created – these are documented on the **Mke2fs** man page. The simplest way, if you are at all unsure, is to look at the **-T** option, but read the documentation carefully before doing anything.

If a lack of inodes is the problem, you can regain some simply by deleting files. Removing a large number of small files is particularly effective.

Q Upgrade upset

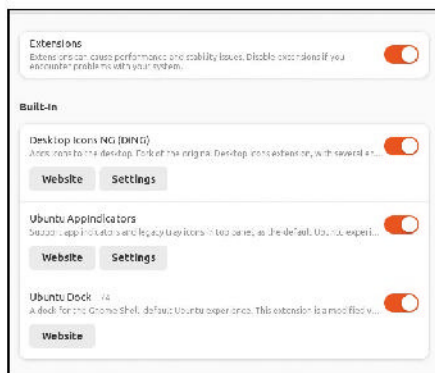
I recently carried out an upgrade from Ubuntu 22.04 to Ubuntu 22.10. The main issue I had was that this time things didn't go quite as before. When the computer restarted, it was a TTY command requesting my username, which I fortunately got correct. I found on Google that someone else had similar problems; their solution was to install Ubuntu Desktop, and my system went to a normal startup screen for Ubuntu.

The only problem was that some of my folder icons, which had been coloured folders, were grey with an exclamation mark inside a triangle. I can click on these and gain access to the subfolders and files inside them. Usually, in previous versions of Ubuntu, you would right-click on a folder and choose a folder colour to change it but the option to do this is missing. I've attached a screenshot, which shows some folders unaffected by the colour change.

Ronald Dennett

A It sounds like your upgrade did not complete fully. If even the standard desktop package was not installed, who knows what else could be missing? You can make sure everything is updated with this terminal command:

```
$ sudo apt dist-upgrade
```



Ubuntu's Extensions program brings some of the settings covered by Gnome Tweaks in house.

Make sure this completes with no errors, or deal with any that crop up. Once you have a clean upgrade, log out and back in, or reboot to be absolutely certain. This fixes the problem in some cases, but others report that it continues after a **dist-upgrade**. In that case, it can be caused by a missing package, so run:

```
$ sudo apt install gnome-shell-extension-desktop-icons
```

You should not need to log out and in again to make these available – just restart the desktop with **Alt+F2+r**. If you still have missing icons, run the **Extensions** app. This is an included program, part of Gnome Shell Extensions, that performs some of the functions provided by the third-party **Gnome Tweaks** software. Check that Desktop Icons is enabled in here and exit.

Q Background business

I'm running a server and want to execute a continuous ping and dump to a file. The script works great at the moment but the problem is when I run the script I cannot return to the console. It keeps running in the console. If I stop it forcefully, the process is also going to stop. If I put an ampersand at the end, can I exit the console and leave the script running and look at it, say, tomorrow?

Kian Gardner

A The ampersand tells the shell to run the program in the background, so you get your prompt back, but the program is still a child of the shell. This means that when you close the shell, your program or script will also be closed. There are a couple of ways to do what you want. The most basic is the **nohup** command, which runs a command in such a way that it ignores HUP (termination) signals, so the following should do what you want:

```
$ nohup yourscript &
```

Any output from the command is sent to **nohup.out** in the current directory. If

that is undesirable, redirect standard output and standard error to **/dev/null**:

```
$ nohup yourscript >/dev/null 2>&1 &
```

This does mean that you are unable to interact with the script after it has started. If that is a problem for you, consider using **Screen** or **Tmux** to run the script in a detachable shell process. Using **Screen**, which you may already have installed:

```
$ screen -m yourscript
```

This starts a screen session running your command. You can detach the screen session from the current shell by pressing **Ctrl-A D** and the script continues. Should you wish to get back to it, simply run:

```
$ screen -r
```

The beauty of this is that you can resume the session from a different shell, even an SSH shell, and you still see your running script. If you just want to start the script without seeing the screen session, add **-d** to the options:

```
$ screen -d -m yourscript
```

Nothing seems to happen, but **Screen** is now running your script in a detached session. You can still reconnect to it with **screen -r**. When you call **Screen** with a command, the session terminates when the command does – this should not be an issue with your use case as it appears your script is meant to run indefinitely.

If you prefer to use **Tmux** instead of **Screen** the equivalent commands to start a session are:

```
$ tmux new ./test.sh
```

or

```
$ tmux new -d ./test.sh
```

The latter version of the command starts the session detached. 

GET HELP NOW!

We'd love to try to answer any questions you send to answers@linuxformat.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 Neil is a pain in the bum), 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 let us know what the exact message was 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
```

Mailserver

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 letters@linuxformat.com.

Print me

A comment on the 3D printing article in **LX296**: it said nothing about Linux; nothing about software needs, supporting libraries, preferred languages, user skill set or levels; and nothing about how open source has fostered the development of 3D printing (only that it has fostered it). And although it provided some history of the development, even that was superficial. It's not evident what the reader is supposed to do with the content of this article.

As a result, it came across too much as just promoting the 3D printing market, not providing a tutorial introduction, an analysis of past or future, or a critical review, which is what **LXF** articles almost always provide.

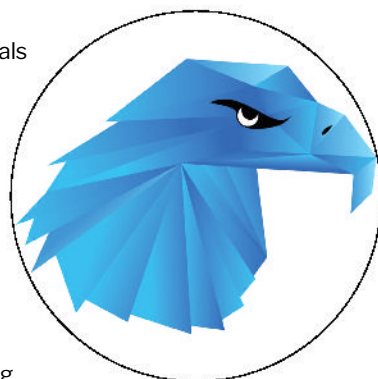
Gregory A Miller, Chicago

Neil says...

I'll throw my hands up and agree with you. In fact, I sent this reply to the author: "If I was concentrating a bit more, I should have perhaps asked you to lean into the open source origins of consumer 3D printing and explore if that's still important and how it's being used ... but the brief was looking at the next-gen tech. So all good!"

The end of that reply is where my thinking ended up, that the article was really to cover where 3D printing tech currently is – rather than the day-to-day practicalities of using it and thus the software side, which I think would be an article in itself. Alastair did a

couple of practical tutorials using 3D printers that covered the software side more fully, but as with so many topics, we could fill an entire magazine just looking at the subject, so it's finding that balance. Thanks again for your feedback – it's very helpful in making the mag.



Another slick Arch distro? They're getting as abundant as Ubuntu spins...

More Manjaro

In **LXF294**, you asked (in a picture caption on the Letters page) what distros we would like to see more coverage on. I would love to see more on both Manjaro and especially Garuda.

Chris Williams

Neil says...

When we ran a DVD (*quiet at the back!-ED*), Manjaro was a regular choice for prime position because it was really popular when it first appeared, managing to bundle Arch into a very easy-to-use and slick package. It almost felt as though it was trying to be the Mint of the Arch world, although more recently there have been a few odd stories around how it's maintained and run. Garuda is a new one to me; looks like it appeared mid-pandemic. At the very least, we'll get a review primed.

Car Format?

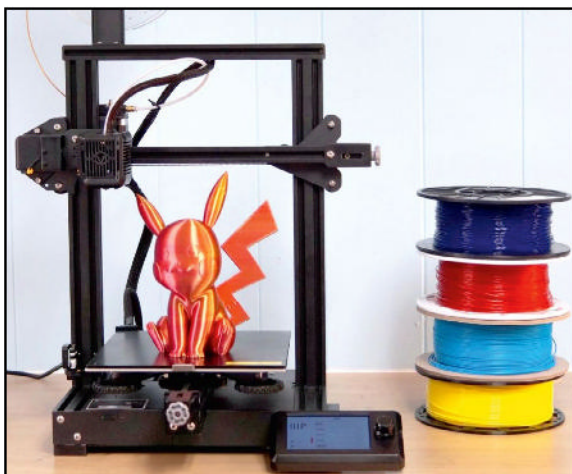
Is there a possibility that you could include an in-depth article regarding software options and use relating to car port OBD readers?

David

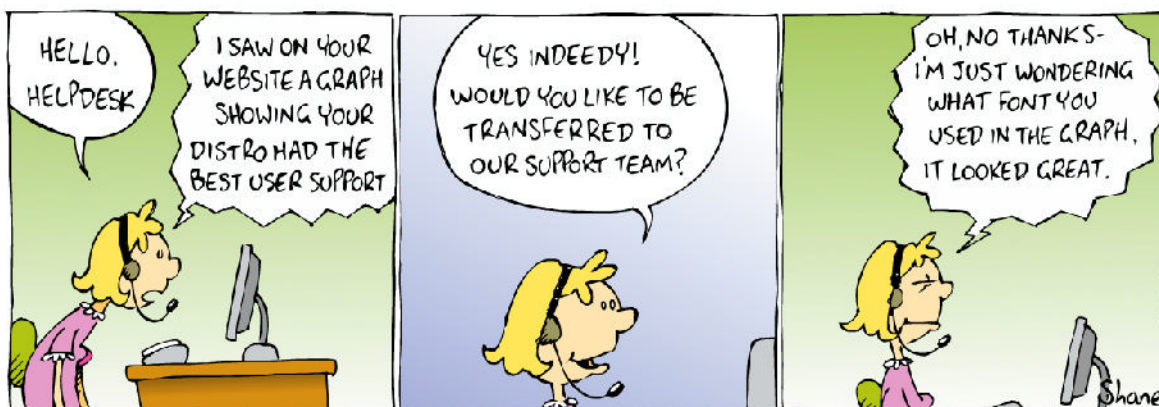
Neil says...

I remember having a conversation with a mechanic around 20 years ago complaining about requiring an expensive dedicated hardened laptop with a bundle of connectors to run engine diagnostics. Looks like there are cheap off-the-shelf Android-based scanners and Bluetooth adaptors now. A quick search returned www.obdtester.com/pyobd but I have no idea whether it's of practical use. It certainly sounds

Did we get too distracted by the shiny hardware? Probably.



Helpdex



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

interesting but my list of Linux-using mechanics is pretty short. Sounds like the sort of crazy tutorial we'd normally run, though, if I can find someone to do it!

We were mainframed!

I was a bit disappointed about the Helpdex in LFX296, December 2022, page 14.

Mainframe systems are usually replaced or upgraded every three or four years; the Cobol compiler every year. That maintenance guy in his 60s has a huge amount of experience and a very high skill set, all of that actually being state of the art. Yes, I know this is a comic strip, but this is so common that people really think mainframes are outdated. This is from the viewpoint of someone who's been maintaining Linux on mainframes since 2000 that are still running salary programs written in 1983!

Tore Agblad

Neil says...

You're not wrong – I don't think most consumers realise how widely mainframes are still utilised. We did run an article on the Linux Foundation's Open Mainframe Project back in LFX266 that was a real eye-opener – including the fact no mainframe has ever been hacked, as far as is known, because of diligent security. As you point out, it's just a cartoon that's poking fun, but I think it also highlights how misunderstood mainframes are. **LFX**



Can Tux diagnose your car engine? We don't know – let's find out!

» LETTER OF THE MONTH

Manifestly wrong

The Manifest v2/v3 article was interesting – all new to me – but the article got a bit carried away with its worship of *Firefox*. It didn't take me long to find that *Firefox* is moving to Manifest v3: <https://blog.mozilla.org/addons/2022/05/18/manifest-v3-in-firefox-recap-next-steps/>



They are implementing it in a different way from *Chrome*, and the function needed for proper ad blocking will stay. At least for the moment. You may remember that *Firefox* broke all extensions a few years back in the name of security and compatibility with other browsers.

This change is also being done for those same reasons, and according to the comments on the above page, some extensions will also end up getting broken.

Speaking of ad blockers, Facebook has got Google to push FBP (the extension that makes Facebook usable) out; thankfully, it is still working with *Firefox*.

You can find out more and help campaign against the change at www.fbpurity.com/news/calling-all-google-chrome-users-please-join-the-campaign-against-google-and-facebook.

Darren

Neil says...

I disagree to an extent – supporting and switching are two different things, I would argue. Mozilla has stated that it'll continue to support Manifest v2 alongside v3, which it's able to do because it's not a *Chromium*-based browser.

But yes, we did get a little carried away in that article, but isn't it nice to get excited about things now and again? However, we should have made the situation with *Firefox* and Manifest v3 clearer, but the main thrust was how *Firefox* helps protect your online privacy.

The previous *Firefox* extension-breaking change was very much needed, because all extensions had complete access to browser resources (I believe), which was a huge security concern.

The Manifest change is more (primarily) about personal data – the argument being that as you're now anonymous, you don't need to block adverts. Though I'm not an expert (in anything), so always happy to be corrected!



shane_collinge@yahoo.com

SUBSCRIBE Save money today!

SUBSCRIBE TO LINUX FORMAT MAGAZINE

**NOT FROM
THE UK?**

Turn to page 65
for more great
subscriber
deals!



Subscribe today and enjoy great savings!

www.magazinesdirect.com/LIN/B23R

Call 0330 333 1113 and Quote B23R

Save money today! **SUBSCRIBE**

1) Only available to www.magazinesdirect.com subscribers

» **PLUS:** Exclusive access¹ 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



DON'T MISS!
Includes 5 years of *Linux User & Developer* issues

» CHOOSE YOUR PACKAGE!

Our simple three-month best offer breaks up the annual subscription into four easily manageable payments over the course of the year.

ANNUAL PRINT EDITION



Only £10.54
every 3 months

13 issues of *Linux Format* in print over 12 months and save 50%!

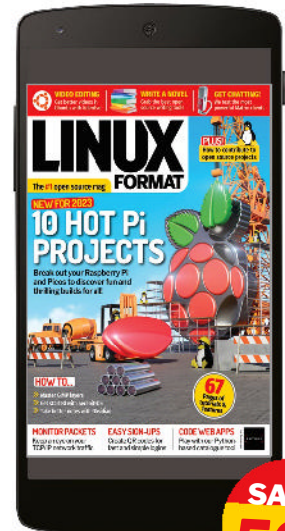
ANNUAL PRINT + DIGITAL EDITION



Only £16.79
every 3 months

13 issues of *Linux Format* over 12 months in print and digital, and save 61%!

DIGITAL EDITION ONLY



Only £11.35
every 3 months

13 issues of digital *Linux Format* over 12 months, and save 50%!

*Terms and conditions: Offer closes 31/3/2023. Offer open to new UK subscribers only. Pricing is guaranteed for the first 12 months and we will notify you in advance of any price changes. Please allow up to six weeks for delivery of your first subscription issue (up to eight weeks overseas). The full subscription rate is for 12 months and includes postage and packaging. If the magazine ordered changes frequency per annum, we will honour the number of issues paid for, not the term of the subscription. For full terms and conditions, visit www.magazinesdirect.com/terms. For enquiries please Telephone: +44(0)330 333 1113 (open Monday-Friday, 8:30am-7pm, and Saturday, 10am-3pm UK time, or email: help@magazinesdirect.com. Calls to 0330 numbers will be charged at no more than a national landline call, and may be included in your phone provider's call bundle.

EXPLORE THE POSSIBILITIES THAT LINUX HAS TO OFFER

From open-source software to coding masterclasses and Raspberry Pi projects, get the most from your machine with this exciting array of expert tutorials, guides and advice from the minds behind Linux Format magazine.



FUTURE



Ordering is easy. Go online at:

magazinesdirect.com

Or get it from selected supermarkets & newsagents

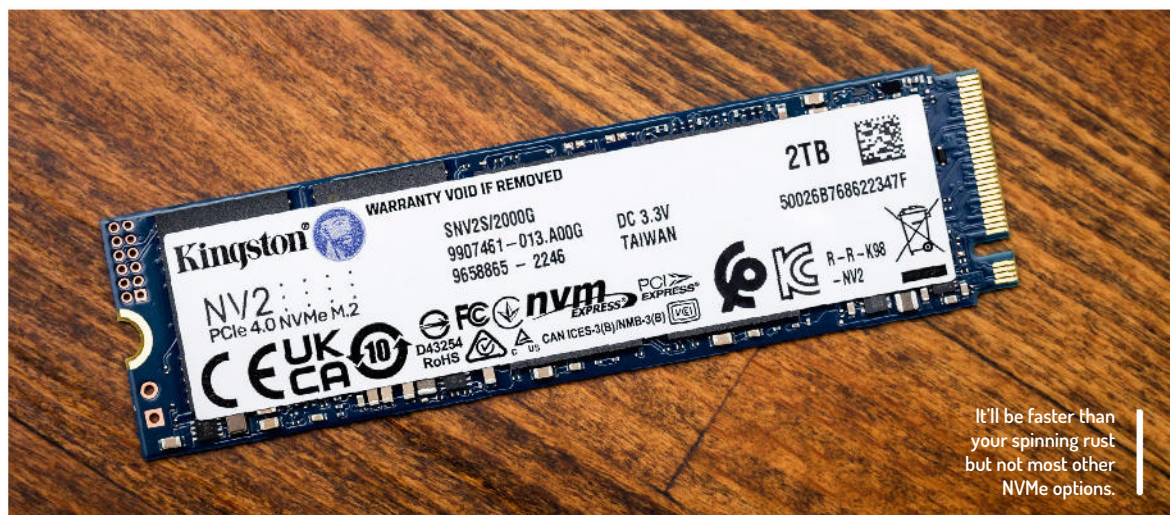
REVIEWS

Kingston NV2

Cheap but not cheerful, bemoans **Shane Downing**.

SPECS

Capacity: 2TB
Format: M.2
2280 80mm
Interface: PCIe
4.0 x4, NVMe
Seq. read/write:
3,500/
2,800MB/s
Endurance:
640TBW
Security: N/A
Warranty:
Three years



The Kingston NV2 is one of those SSDs that is simply too good to be true. The price is exceptional, especially at 2TB, and it is sold as a PCIe 4.0 drive. Yet it has no definitive hardware inside and its performance for both of our samples is distinctly bottom of the barrel. It also runs a little hot and inefficiently in our testing.

The NV2 follows Kingston's previous NV1, a drive very much in line with Kingston's philosophy of providing cheap drives at scale. The NV2 is similar in that it uses a hodge-podge of hardware – different controllers and NAND flash from drive to drive – offered at an insanely low price. The Kingston NV2 is available in 250GB, 500GB, 1TB and 2TB flavours.

The drive can manage up to 3,500MB/s and 2,800MB/s for sequential read and write, respectively, but has no random performance specifications. This makes sense because it can utilise more than one controller and more than one type of flash. The sequential write specifications are such that it can only have QLC at 1TB and 2TB, however. The sequential values are low for a PCIe 4.0 drive for a good reason: Kingston set them for the weakest possible controller and flash.

The NV2 has a three-year warranty and can manage 320TB of writes per TB capacity. This is exactly as expected for a budget drive.

The 2TB model we sampled is using the SMI SM2267XT controller. This is one of those entry-level PCIe 4.0 SSD controllers that barely qualifies for the 4.0 moniker. It's similar to the Phison E19T controller used on drives such as the WD Black SN750 SE or Inland TN436. This technology has a 1,200MT/s bus, which, with four channels, means it can saturate a x4 PCIe 3.0 link, much like the controllers on the SK Hynix Gold P31 and WD Blue SN570. For all purposes, this makes the controller a jacked-up SM2263XT with

higher bandwidth and IOPS, but not really new technology, such as the SM2269XT.

Our 1TB sample also uses the SM2267XT, but this drive has been spotted with the newer SM2269XT controller as well. Given the limited performance specifications, it's possible this drive could also come with the comparable E19T, too.

In testing, the drive also ran hot, even when idle, and it draws a decent amount of power, making it less than ideal for many machines, especially laptops, which is likely a prime destination. Its saving grace is its amazing price, which does make it perhaps good for something other than a primary drive. It often came dead last – by a good margin – in our 50GB file copy both in terms of time and MB/s per watt, while it was usually vying for last place across the board.

Our samples both use the SMI SM2267XT controller, which did not perform well with either TLC or QLC. The flash, in either case, is actually not bad. It's just that there are so many better options currently on the market. **LXF**

VERDICT

DEVELOPER: Kingston

WEB: www.kingston.com/ssd

PRICE: £110 2TB (£53 1TB, £34 500GB)

FEATURES	7/10	EASE OF USE	8/10
PERFORMANCE	4/10	VALUE	7/10

The 2TB Kingston NV2 is a dirt cheap NVMe SSD and not much more. Performance is poor, the drive runs hot, and you cannot be certain of the hardware.

» **Rating 5/10**

AMD Ryzen 7950X

Dave James pens these words and takes complete blame for saying: Zen 4 and the art of CPU maintenance.

SPECS

Socket: AM5
Process: 5nm TSMC
Cores: 16
Threads: 32
Clock: 4.5GHz (5.7GHz boost)
Cache: 1MB L1, 16MB L2, 64MB L3
Unlocked: Yes
GPU: AMD Radeon 2-core RDNA 2, 400MHz
Mem max: 128GB DDR5-5200, ECC support, 2-channel
PCIe: v5 24-lanes
TDP: 170W.

The AMD Ryzen 9 7950X is the vanguard for a whole new generation of CPUs from AMD. At its most basic level, this is a straight 16-core, 32-thread CPU, packing a pair of eight-core compute chiplets together with an I/O die to get all the data into and out of the processor package.

Zen 4 is a derivative architecture built from Zen 3 with fundamental differences: it's built on a new 5nm process node from its long-time TSMC foundry partner, while the I/O die (IOD) has shifted to a 6nm node; AMD has made the move to a DDR5 memory controller, and it's also popped PCIe 5.0 support in there, too. For the first time outside of its APU range, AMD is also packing integrated graphics into its chips, into the IOD that offers AV1 hardware support and hybrid graphics. You will want a recent Kernel 5.18+, Mesa 22.2+ with the latest firmware to get this working.

Architecturally, Zen 4 is a similar layout to Zen 3 but with 'more' for the critical areas that keep execution units fed – it has double the L1 cache to reduce latency, enhanced branch prediction, larger op cache, retire queue, register file, buffers throughout and enhanced load/store for a 60% increase in IPC.

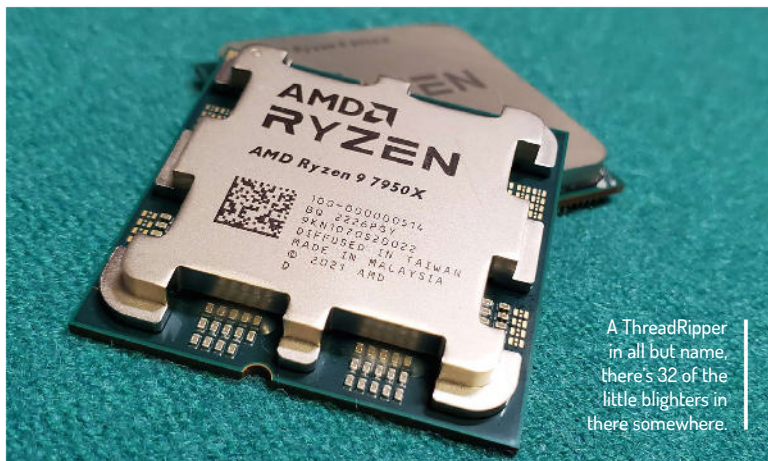
Getting up to speed

It is certainly fast. We did a bit of a double take checking out the numbers when the single-thread run tipped the odd core up to 5.9GHz. Even stable at 5.8GHz was pretty shocking. It's not quite that high under an all-core load, but 5.4GHz across all 16 cores still makes it one blazing chip.

For pure productivity, the 16 cores are formidable. In the *Blender* render test, the Ryzen 9 7950X is over 50% faster than the Core i9 12900K, and over 40% faster than the Zen 3-based Ryzen 9 5950X. That highlights just what a multithreaded monster the new Zen 4 platform can create, and is where all those front-end improvements really come to the fore. Though for single-thread speeds, the Intel Core i9 is still dominant and even the Ryzen 7700X is ahead.

It's not the same story for gaming performance, if that's important. Across the board, the Ryzen 7950X slightly lags the Intel Core i9 12900KS but also usually the Ryzen 7700X. This doesn't make for great reading when you consider that it will soon be replaced by even faster Raptor Lake processors that could push gaming performance even further ahead of AMD's chips.

If you're happy digging around BIOS settings, AMD has an Eco mode that locks power draw to just 65W,



A ThreadRipper in all but name, there's 32 of the little blighters in there somewhere.

rather than the peaks of 191W it could consume. The result is just the loss of a few frames per second, less heat and a quieter system that will still out-render an Intel Core i9 12900K.

The new AM5 platform's penchant for the number five is demonstrated by the increase in memory bandwidth owing to the support for DDR5 memory. With the new EXPO memory profiles, we could instantly hit the top frequency with our G.Skill DDR5-6000 kit. That delivers an easy 68% increase in memory bandwidth over the previous Zen 3 generation. The benefits of the PCIe 5 interface, however, won't manifest until we get hold of some solid-state storage drives maxing out the increased bandwidth when they arrive later this year.

This £600 16-core processor, despite its heady 5.8GHz clock speed, still falls behind Intel's slightly cheaper Core i9 12900K when it comes to raw frame rates. And you can still get the GPU-less KF version of that Alder Lake CPU for much less than that. But the multicore performance is staggering – there's nothing in this price bracket that can come close to the processing prowess of AMD's new Zen 4 top chip. **LXF**

VERDICT

DEVELOPER: AMD

WEB: www.amd.com/ryzen

PRICE: £600

FEATURES	9/10
PERFORMANCE	8/10

EASE OF USE	9/10
VALUE	8/10

A multithreading monster of a computational CPU, hitting speeds we previously only got near with serious overclocking.

➤ **Rating 8/10**

Rocky 9.1

Nate Drake goes head to head with Red Hat-based distros this month, so is glad to have the heavyweight Rocky in his corner.

IN BRIEF

One of the new CentOS alternatives that has a strong commercial backing and is headed by an original CentOS developer.

SPECS

CPU: 2GHz (single-core)
Mem: 2GB
HDD: 10GB
Builds: x86-64, aarch64, ppc64le, s390x

At *Linux Format*, we're very into Debian-based Linux, but don't want you to think that we've anything against RPM distros. One of the best known is Red Hat Enterprise Linux (RHEL), a commercial version of Linux.

This may sound counterintuitive, as the whole ethos of Linux is one of free and open source software. Red Hat does make its source code publicly available but also sells subscriptions and training. Other developers can use RHEL's source code to make their own distros, as is the case with Rocky 9.1

Rocky Linux was originally pioneered by one of the founders of CentOS, Gregory Kurtzer. After Red Hat announced last year that it wouldn't continue development of CentOS, the community-supported free version of RHEL, the team at Rocky stepped up to create this binary compatible release.

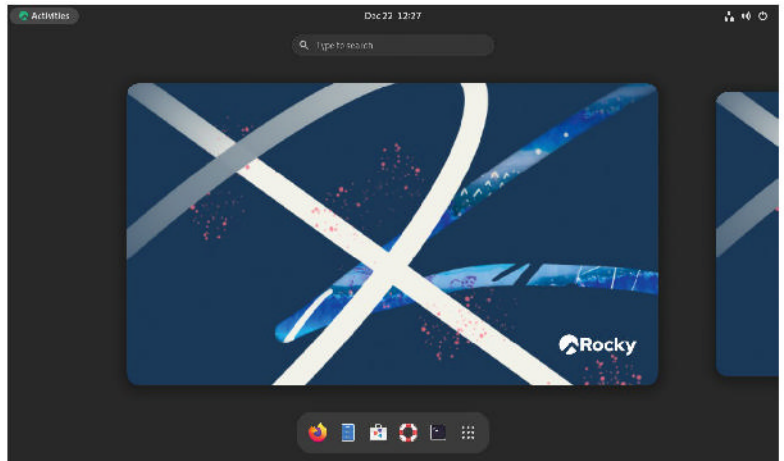
The most recent result, Rocky 9.1 Blue Onyx, is an extremely reliable OS. Being based on RHEL means that it has been checked thoroughly to make sure it's stable and functional. In the release notes, the team states this version went through nine days of testing, carefully documented on the website. In theory, this means you're benefiting from effectively the same OS for which others are paying, with support until 2032.

Weight watching

In the nature of things, though, this means there's a colossal number of packages available. The regular install DVD weighed in at over 9GB but there are various download options, including via BitTorrent. You can also choose to download the 1.5GB Minimal ISO, then install the GUI and other packages via the command line, as we did. This is all helpfully explained in the Rocky Linux online documentation.

Sadly, there's no way to test Rocky in Live mode but setup itself is a breeze. The installer gently guides you to choose your username, time zone and target drive using a simple GUI. We were also very impressed that setup makes you set a root password, as this stops bad actors from trying to do this post-install.

When your desktop loads for the first time, setup also prompts you to connect your online accounts and set your own username and password. You're then treated to a whistle-stop tour of the Gnome 3 desktop. The Gnome Software Centre is just as easily able to install RPM programs, among which you find favourites such as the *Thunderbird* email client if you want an alternative to the pre-installed *Evolution*. The VLC



The flagship version of Rocky 9 uses Gnome 3 but other desktop environments are available for download.

media player is notably absent by default but you can add this and other programs by enabling community-based repos such as EPEL (Extra Packages for Enterprise Linux) and RPM Fusion, then installing via the command line. If Gnome 3 isn't to your taste, the Rocky Linux website also offers ISOs with the Gnome Lite, KDE, Xfce and MATE desktop environments.

Since CentOS was axed, Rocky Linux has gone from strength to strength. According to EPEL stats, it's the most popular distro of this kind. In a Reddit chat last year, the Rocky team said this may be because the community is much larger than that behind CentOS. The devs have also released the open source build stack Peridot, so anyone can build new versions of the OS and it's very unlikely to share CentOS's fate.

This should make your decision whether or not to switch to Rocky very simple: if you want to benefit from the stability and support of Red Hat Linux without the price tag, the people have spoken. Rocky offers the same functionality, is easy to set up and has a huge amount of online support. **LXF**

VERDICT

DEVELOPER: Rocky Enterprise Software Foundation

WEB: <https://rockylinux.org>

LICENCE: 3-Clause BSD and others

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

If you're a Red Hat fanatic, Rocky is an ideal OS to set up and use, with easy installation and good online support.

» Rating 8/10

Linux Mint 21.1 Vera

Nate Drake meets Linux Mint's latest offering, code-named Vera, and comes away bowled over by its stunning looks and ease of use.

IN BRIEF

Starting as an Ubuntu spin-off, Mint has grown and matured with its own desktop direction and suite of easy-to-use system tools.

SPECS

CPU: 64-bit (single-core) 2GHz
Mem: 2GB (4GB recommended)
HDD: 20GB (100GB recommended)
Builds: x86_64

Unless you've been living under a rock these past 15 years, you'll almost certainly have heard of Linux Mint. The OS is based on Ubuntu (in the case of this release, Ubuntu 22.04 LTS), meaning you have access to almost all Ubuntu applications out of the box.

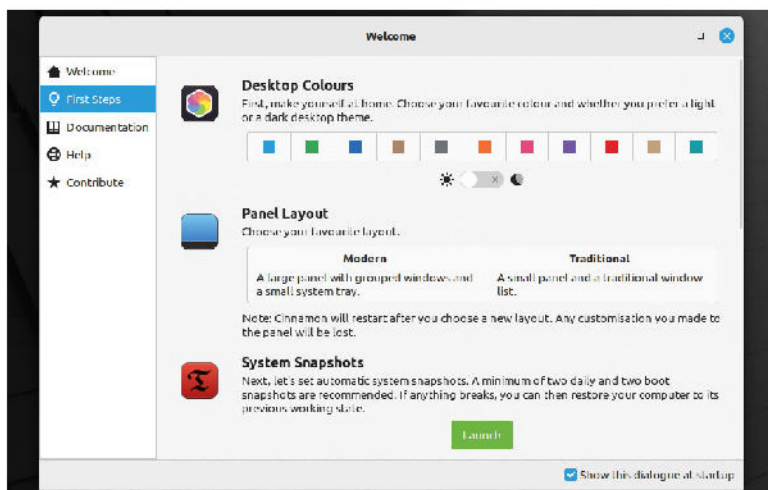
"Out of the box" should perhaps be the tagline for Linux Mint, as the developers have placed a real focus on ease of setup and deployment for desktop users. Mint comes bundled with the ability to play almost all media formats. Traditionally, this has included proprietary software such as *Adobe Flash Player* but FOSS enthusiasts can take comfort in the fact that the Mint installer now offers these on an opt-in basis.

Mint has a respectable choice of built-in apps. Productivity is a breeze with the *LibreOffice Suite* (7.3.7.2) and the *Thunderbird* email client (102.4.2), while media needs are catered for via the powerful *Rhythmbox* (3.4.4) and *Celluloid* (0.21) music and video players.

You may notice that these aren't necessarily the most recent versions of these programs. This is because the Linux Mint team monitors software updates carefully, making for a much more reliable system. As Vera is based on an LTS version of Ubuntu, it should not only be more stable but will remain supported until 2027.

You can also use the included Mint software manager (*MintInstall*) to add extra apps. The program obviously draws on *GNOME Software* for inspiration, so is clear and easy to navigate. By default, it supports installation of apps via both Flatpak and DEB. If a program can be installed in more than one way, you can now choose between them.

Mint's cool features are simply explained as you go through the installation process but, in keeping with the devs' focus on ease of use, on first boot you're also presented with a welcome guide, which prompts you to choose your desktop colours, light or dark theme, and panel layout. Colour is a mainstay of this most recent



Get started by selecting your own vibrant desktop colours and panel layout. There's also a huge number of desktop backgrounds to choose from.

version of Linux Mint, as the devs have been at pains to point out that system colours are now far more vibrant.

Mint condition

Linux Mint 21.1 also now uses its own variation of the Aqua theme, a departure from previous versions where folders and icons were a minty green. If yellow folders aren't to your taste, the old Mint-Y-Legacy theme is still available from Preferences > Themes, as are the Breeze, Papirus, Numix and Yaru themes. You can also come here if the new Bibata mouse pointer isn't to your taste.

There's a huge selection of desktop backgrounds available, both from Linux Mint 21 Vanessa and the most recent Vera 21.1 release.

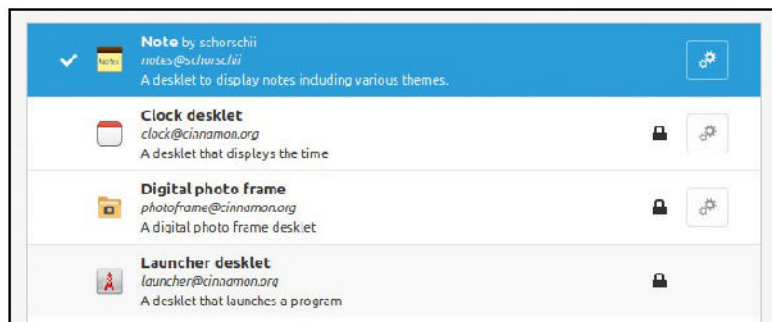
Once you're done with cosmetics, you're asked to set up automatic Snapshots, so if anything goes wrong with your machine, you can roll it back to a working state, like Windows System Restore. This is done via *Timeshift*, a GUI front-end for backups via *RSync* or *BTRFS*.

This is where Mint really comes into its own, because while it is possible to make recursive backups in regular Ubuntu, users aren't prompted to do this out of the box using the OS's pre-installed tools.

The Welcome guide also points new users to the extensive online documentation, web forums and chat room available for Linux Mint.

The flagship version of Linux Mint 21.1, which we're reviewing here, uses the Cinnamon desktop environment, a fork of GNOME Shell. The release notes for Mint 21.1 state that the environment has a "fresh look and feel" and we can't help but agree – by default, the desktop only shows the bare essentials: an app launcher plus a handful of icons for the *Firefox* browser (108.0.1).

You can also click the Files icon to launch the *Nemo* file manager. By default, this opens your **home** folder, so the desktop icon has been removed. You need to launch



» LINUX MINT ALLSORTS

We've focused on the Cinnamon version of Mint 21.1 but there's also an official release using the MATE desktop environment, which is based on Gnome 2. MATE is less resource-hungry than Cinnamon, so might be better for older machines or to get more out of your laptop battery.

Although it can't use all Cinnamon features, such as Desklets, MATE is extremely stable and can run with just 2GB of RAM. Visually, Mint Cinnamon and MATE are very similar, though the MATE app menu and windows look more dated. Linux Mint MATE also uses the *Caja* file manager over Cinnamon's *Nemo*, though both are based on *Nautilus* and are readily configurable.

There's also an official build of Linux Mint using Xfce. This has a similar low resource footprint to MATE and is extremely customisable. It isn't updated as often as the other desktop distros but this can be an advantage if you value stability over ultra-modern apps and features.

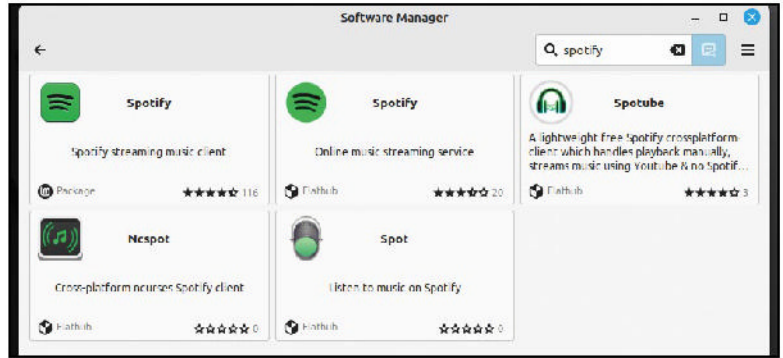
If you prize integrity above all, consider LMDE – the most recent version, 5, is designed to be as similar to regular Linux Mint as possible but uses Debian as a package base (in this case, Debian Bullseye), ensuring extra stability and long-term support. LMDE still supports 32-bit architectures (unlike mainstream Mint). By default, it uses a slightly older version of the Cinnamon desktop.

Visit https://linuxmint.com/download_all.php to see all the available versions of Linux Mint.

Nemo to access Trash, Network and Computer as the icons for these are gone, too, but as the Mint team notes, you can re-enable them in System Settings > Desktop.

Cinnamon also supports Desklets, which can be placed on your desktop to add more features. The pre-installed options are fairly basic, only enabling you to launch programs, but there are many community-developed ones available, such as *Notes*, which places virtual sticky notes on your desktop.

Cinnamon has a more extensive selection of built-in Applets, which you can view from Preferences > Applets. These are mainly designed for the panel and allow you to customise your desktop even further. We're particularly



impressed by the *Recent Documents* Applet, which no doubt will save much hassle. Vera also makes navigation easier by introducing a Show Desktop button at the bottom-right to quickly dismiss all windows.

Although we've focused on the Cinnamon distro of Linux Mint, the OS is also available for download with the MATE and Xfce desktop environments. This might come as a relief if you prefer a more traditional desktop, without needing your system to be infinitely configurable (see *Linux Mint Allsorts boxout*).

Special mention should also go to Linux Mint's update manager, which even power users may enjoy. While in theory you can update with a line of text from Linux's CLI, using *MintUpdate* gives you a great deal of control.

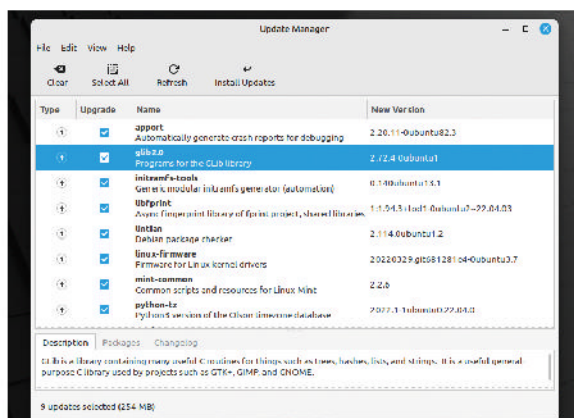
On first run, *MintUpdate* asks if you want to use a local mirror and even pings nearby servers to show you which is fastest. You can then view available updates and click to see a description each. You can even untick those you don't want to apply.

Linux Mint 21.1 is only available for 64-bit architecture, though the offshoot LMDE (Linux Mint Debian Edition) still supports 32-bit machines. There's also no official version for ARM64 processors. This is bad news for anyone who wants to run Linux Mint on a legacy PC, Raspberry Pi or M-Series Mac.

This is a distro for desktop users, reflected in the richness of its graphics and the heavy recommended install footprint of 100GB (though we note that the Linux Mint 21.1 Cinnamon ISO is only 2.5GB and, once installed, only took up around 12GB of space).

The only real criticism to level against Mint is that there are many Linux users who don't require the level of hand-holding it offers in terms of extensive support and friendly GUIs for software installation and updates. Still, even experienced users can benefit from the awesome Cinnamon desktop and the customisability of this OS, as well as how many features work out of the box. **LF**

The Software Manager now supports installing programs both via system DEB packages and Flatpak. Simply choose your method and click Install.



The Update Manager shows all available updates in one place. Click for more details or to only update certain programs.

VERDICT

DEVELOPER: Clément Lefèbvre

WEB: www.linuxmint.com

LICENCE: Various

FEATURES **9/10**

PERFORMANCE **8/10**

EASE OF USE **10/10**

DOCUMENTATION **9/10**

Idly cool, Linux Mint takes you by the hand and guides you into the most user-friendly of operating systems.

» **Rating 9/10**

Return to Monkey Island

The Management thinks three decades is a perfectly fine amount of time for **Will Freeman** to wait to get an under-inflation pay rise.

SPECS

Minimum
OS: Ubuntu
20.04 64-bit
CPU: Quad-core
64-bit
Mem: 8GB
GPU: 2GB
VRAM, Vulkan
1.2 or OpenGL
4.2 support

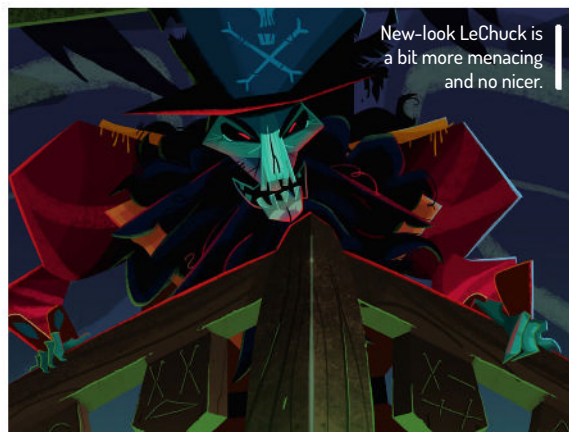
The infamously bewildering ending to 1991's *Monkey Island 2: LeChuck's Revenge* left players longing for resolution that subsequent *Monkey Island* games never delivered. Three decades later, and after effectively saying it would never happen, key members of the original team –including series creator Ron Gilbert –have made the game that picks up after the final moments of *LeChuck's Revenge*. And they subvert our 30-year-old expectations about what happens next immediately.

Return to Monkey Island's mischievous opening confidently asserts that this is the authentic *Monkey Island* experience we've come for: sharp, self-aware and brilliantly silly. It bombards you with gags but the characters you meet aren't just comedic props, and the cunning stream of interwoven puzzles has been modernised to keep up the pace without losing the satisfaction of problem solving. It's a massive success.

The opening prologue whips you through a homage to classic *Monkey Island* moments before thrusting the shambling Threepwood back into his past. Older, tired and more challenged by holding his breath, Guybrush has grown with us, as have *Monkey Island's* classic locales and characters. Everything and everyone has gone on a journey. Wally the cartographer has been reshaped by finding his confidence without losing his charming spirit, and even LeChuck seems to have found a softer side despite remaining the furiously irritated villain we adore.

Everything is different but familiar, from the narrative and characters to the revised systems for interaction and puzzle solving. It can temporarily transport you to the '90s – at least if you also have cosy memories of staying up too late solving pirate puzzles – but it isn't stuck there.

The story is consistently engrossing and energetic, but spoiling too many details would be beneath LeChuck himself, so let's keep things general. After starting out in the first game's Mêle Island, Guybrush takes to the seas to explore vintage locations such as Monkey Island itself, as well as entirely new lands, including the tormented,



eerie Terror Island and Brrr Muda, a place inspired by the Viking age. Guybrush is once more in pursuit of Monkey Island's infamous secret. LeChuck, too, has returned to the quest. And both, of course, remain besotted by Elaine Marley. Neither, meanwhile, has gained any real sense of what the secret of Monkey Island actually is.

All that is the same, but with the passage of time, new figures have risen to power, upsetting the conventions of Guybrush's world. The old guard of established sea dogs now exists on the fringes, replaced by youthful upstarts eager to reset the rulebook. Even voodoo is no longer in favour, with younger generations favouring "dark magic".

New Ch-ARR-acters

A great many of *Return to Monkey Island's* best moments come from the new characters, who are almost universally brilliant. Highlights include the oddly charismatic Putra, undead galley chef to LeChuck, who serves both rasping sass and all manner of delightfully gross food gag setups as she gradually develops an affection for Guybrush. Or there's Mêle's new locksmith, Locke Smith. Deliberately absurd name aside, Locke is stern and cool to the bone, and endures Guybrush at best, delivering all kinds of opportunities for laughs as she dismisses the protagonist's puns about locks of hair, or extinguishes his enthusiasm for treasure hunting as he is bewildered by a map lacking any X to mark a spot.

This is a game constructed from the familiar as much as the new, of course. A lot of *Return to Monkey Island* is built from settings that have undergone a subtle revolution. Visit this game's version of the Scumm bar and everything seems the same. The overall composition is unchanged, right down to where people are seated.

The general design hasn't evolved much – this is a straightforward point-and-click adventure game that plays just as genre devotees expect. Conversing with the game's cast delivers the story, nudges you in the right direction, and seeds your mind with goals and hints, guiding you through puzzles that see you variously hunt down keys, outwit those who stand in your way, extract





information from the environments, and unpick a path to whatever Monkey Island's secret might ultimately be.

Many of *Return*'s finest puzzles are those anchored primarily in conversation. One of the most memorable asks that you learn to spin a good pirating tale by determinedly reeling off badly delivered stories to unimpressed characters across the game world. Their feedback helps you hone your craft: a puzzle about writing in a game that is all about its writing.

X marks nothing!

Compared to the previous *Monkey Island* games, though, the puzzles here play out a little faster. It's not that this is profoundly easier. Rather, you're less likely to get stuck on individual snarls or weird leaps of logic.

If you just want to enjoy the world and humour, there's a Casual mode that simplifies the puzzles and emphasises the narrative. The more significant change, however, is the addition of contextual action descriptions. Hover over interactive objects, and the description of what you can do provides much more flavour than in past games. Hover over the chef's mop, for example, and 'Admire the mop' appears as an option for the left mouse button, while 'Surreptitiously abscond with the mop' is displayed with a symbol for the right-click.

It's a subtle tweak, but the additional text does much more than simply inject more delightful personality, tone and laughs into a game that's already rich in those things. The extra context doesn't make puzzle solving any easier, but it does provide more explicit information on where clicks might lead.

Then there's the hint system. It can be ignored if you who want the satisfaction of getting yourself unstuck, but if you would normally head to Google, the in-game



Characters old and new appear in the latest game.

hints maintain immersion in the world. It's much better than pulling up an external guide, and you can choose how subtle or explicit the clues you digest are.

The most obvious and contentious modernisation is the art. Yes, it's not pixelated. But this is a *Monkey Island* for today, and the look suits Gilbert's world. Almost every screen bristles with more to see than is first apparent – incidental animated background moments, such as busy columns of ants in the undergrowth, for example – making each island feel like a living place. The occasional shooting star feels like a detail from a '90s game, but looks contemporary. There are some gorgeous lighting effects and the scenes are framed exquisitely.

The audio, too, is remarkable, reworking the classic score with variety and energy. The sound effects – from muttered conversations just off-screen to audiobeds rich in detail – are equally wonderful, creating a sense of space in the 2D scenes.

If you haven't already played the other *Monkey Island* games, however, you may want to heed the implication of the title. You don't need to get the in-jokes to enjoy the story, and there's a surprisingly engrossing interactive scrapbook that provides a lively summary of the series's previous events. But to really enjoy *Return to Monkey Island* is to titter at the self-referential gags, or roll your eyes knowingly at Guybrush's deliberately bad lines and choices.

Walking into a familiar building to find an unexpected interior is only delightful if you know the existing lore. Half the fun is in getting the jokes or references before they're fully revealed. As a *Monkey Island* fan, it's hard to imagine what a newcomer's experience would be like. But for lifelong followers of Guybrush's misadventures, this is as good as it's been for a very long time. **LXF**



VERDICT

DEVELOPER: Terrible Toybox

WEB: <https://returntomonkeyisland.com>

PRICE: £20

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

Funny, superbly paced and packed with ideas and eccentric characters. What we've been waiting for since LeChuck and Guybrush emerged from... Well, if you know, you know.

» **Rating 9/10**

Roundup

Flatpak » Source code packages
» Traditional packages » Apptainer » Snap



Michael Reed
installed so many
packages this
month that we
nearly told him
to pack it in.

Packaging formats

There's a number of ways of getting software on to your Linux-powered computer, and **Michael Reed** is comparing them to find out which is best.

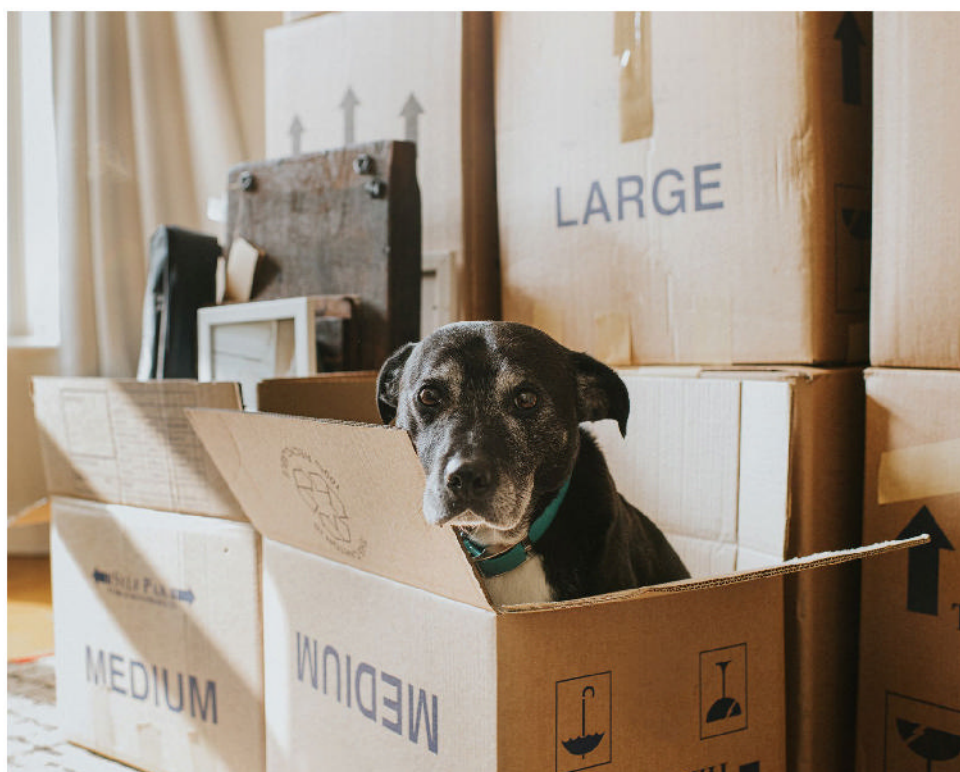
HOW WE TESTED...

We've decided to focus on application installation, but we haven't ignored the installation of system components, and we've given that subject its own section.

We looked at the process of making a package, but we found that the procedures for standard, traditional packages such as RPM, Apptainer, Flatpak and Snap were similar. None of them was really beginner-friendly, but we wouldn't expect them to be.

We searched the repositories for apps such as *GIMP* and *Kdenlive* to get a sense of the relative freshness of packages available in each format. We also took into account that some formats are more popular than others on the websites of open source projects.

When examining traditional package managers such as *RPM* and *DEB*, we've had to give an aggregate assessment as there is some variation between them in terms of features. However, all of the mainstream ones basically, accomplish the same job and go about it in the same way.



When it comes to adding software, Linux has always taken a different tack from other mainstream operating systems. Generally, a packaging system consists of a package format and a set of tools for managing those packages. Practically every Linux distro includes a built-in package manager that automatically retrieves and installs packages in formats such as DEB and RPM. Because traditional package managers are similar in how they work, we've grouped them together here.

Like the system package management tools, *Snap* is a package management

system that can draw packages in its own format from an online repository, and the same can be said for *Flatpak*. Both aim for distribution independence and offer useful sandboxing and security features, and support downloaded packages that can be installed outside of the repository system.

Compiling the source code for a program and installing the result is another classic approach favoured by the more technically inclined Linux user, and we consider that, too.

Finally, *Apptainer* is particularly at home when it comes to supplying complete apps without altering the underlying system.

Basic tools and utilities

A robust feature set is top of our list.

We start our comparison by examining the facilities that each system has to offer when carrying out the basic operations, such as installing and updating packages. The GUI has its place, but we were particularly keen to see how well they all worked from the command line.

There's a slight problem of fragmentation, even within a single distribution, when it comes to traditional package managers as they often use different commands for different operations. OpenSUSE's *Zypper* is an example of an admin utility that ties everything together, but they're not all like that.

Searching for a package and installing it from the command line also works well when using a traditional package manager front-end such as *APT* under a distro such as Debian or Ubuntu. One problem with traditional package managers is that they tend not to distinguish between system components and apps, so searches for apps can be complicated. *Flatpak*'s searches suffer from this, to a lesser extent, and a simple search can throw up lots of packages, leaving you confused about how to proceed. *Snap*'s searches for apps that are available or already installed tend to result in concise lists that are easy to understand.

Fetching source code packages from a distro's repository works in a similar way to fetching normal packages, but installing them can involve a lot of work and research. Doubly so if it's an up-to-date source code package downloaded from a website.

ApplImage takes a different approach from *Snap* and *Flatpak* by not using a central database. However, *ApplImage* archives do

Name	Description	Application ID	Version	Branch	Remotes
GNU Im...	Create images and edit photog...	org.gimp.GIMP	2.10.32	stable	flathub
GIMP U...	GIMP User Manual	...gimp.GIMP.Manual	2.10	2.10	flathub
Resynt...	Set of GIMP plug-ins that hea...	...gin.Resynthesizer	2.0.3	2-40	flathub
Resynt...	Set of GIMP plug-ins that hea...	...gin.Resynthesizer	2.0.3	2-3.36	flathub
Gimble...	Gimblefun is a Gimp plugin	...MP.Plugin.Lensfun	0.2.4	2-40	flathub
Gimble...	Gimblefun is a Gimp plugin	...MP.Plugin.Lensfun	0.2.4	2-3.36	flathub
Fourier	A simple GIMP plug-in to do f...	...MP.Plugin.Fourier	0.4.3	2-40	flathub
Fourier	A simple GIMP plug-in to do f...	...MP.Plugin.Fourier	0.4.3	2-3.36	flathub
BIMP	Batch Image Manipulation Prog...	...GIMP.Plugin.BIMP	2.6	2-40	flathub
BIMP	Batch Image Manipulation Prog...	...GIMP.Plugin.BIMP	2.5	2-3.36	flathub
Liquid...	LiquidRescale plugin to resiz...	...gin.LiquidRescale	0.7.2	2-40	flathub
Liquid...	LiquidRescale plugin to resiz...	...gin.LiquidRescale	0.7.2	2-3.36	flathub
G'MIC	GREYC's Magic for Image Compu...	...GIMP.Plugin.GMIC	3.1.6	2-40	flathub
G'MIC	GREYC's Magic for Image Compu...	...GIMP.Plugin.GMIC	2.9.6	2-3.36	flathub
FocusB...	Focus Blur plug-in crete a bl...	...Plugin.FocusBlur	3.2.6	2-40	flathub
FocusB...	Focus Blur plug-in crete a bl...	...Plugin.FocusBlur	3.2.6	2-3.36	flathub
Glimpse	Create images and edit photog...	...se_editor.Glimpse	0.2.0	stable	flathub
Swatch	Color palette manager	org.gabmus.swatch	0.1	stable	flathub

The output of "flatpak search gimp". As you see, it's a bit confusing, and a newcomer to the system would be at a loss for what packages are needed to install GIMP.

have a mechanism for searching for updates. The problems are that packagers have to implement it themselves, it's not widely present in *ApplImages* and it can't do system-wide updates.

Update features tend to be very good when using the built-in package manager of a distro. As often as not, a single command updates the package database, and another updates the packages themselves. *Snap* and *Flatpak* combine the operations into a single command. By default, *Snap* is configured to carry out regular updates in the background, which is annoying as this can consume a great deal of CPU and network resources. This behaviour can be changed from the command line, though.

VERDICT

FLATPAK	8/10	APPMIMAGE	6/10
SOURCE CODE PACKAGES	5/10	SNAP	8/10
TRADITIONAL PACKAGES	7/10		

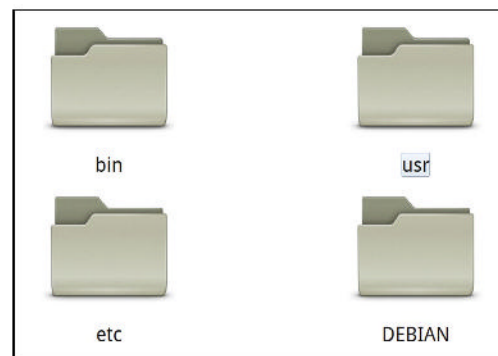
As ever, the traditional package managers keep the system running, but *Flatpak* and *Snap* offer some refinements.

Portability of your packages

Are packages fixed to one distro?

System package formats (such as RPM and DEB) are linked to the Linux distro they were created on. There are tools for converting from one type to another, but we're not taking those into account as they are not fully reliable. Even between distros from the same family, traditional packages are not reliably portable. For example, a DEB archive created on Debian may work on Ubuntu, but there's no guarantee; even different versions of the same distro might be incompatible at package level. Portability of system package formats is poor.

Snap is widely supported, particularly in the mainstream distributions. If it's not installed by default, it can generally be added, but that does involve an extra stage for the user, and not every small distribution supports *Snap*. Much the same can be said for *Flatpak* as it does need support in the distribution. *Flatpak* is slightly better supported than *Snap* on distributions that are dissimilar to Ubuntu because it requires *Systemd* to work and *AppArmor* for full sandboxing. Even though it is similar to Ubuntu, Linux Mint is an example of a distribution that makes installing the *Snap* system difficult.



A package format such as DEB is just an archive with program and control files. It's efficient, but the target system has to have the libraries that the program needs.

As long as a given distribution provides a reasonably standard environment, *ApplImage* archives don't require any support from the system itself. Nine times out of 10, even if it's a small, niche distribution, *ApplImage* installation should work, and that's also a boon in situations where you don't want to have to add extra packages to a setup just to be able to add extra applications.

VERDICT

FLATPAK	8/10	APPMIMAGE	9/10
SOURCE CODE PACKAGES	6/10	SNAP	7/10
TRADITIONAL PACKAGES	6/10		

ApplImage is the portability king. It can even run from a read-only medium while storing application settings within the home directory.

Repositories and shops

Locating and adding apps from the GUI.

In keeping with this *Roundup* in general, we're placing a greater emphasis on the installation of applications rather than system components particularly.

Reviews and ratings are important in this respect. They can give you a heads-up if a current release exhibits regressions, and they usually contain comments about the implementation of that package within that particular packaging system.

It's worth considering the packaging system when evaluating a Linux distribution. The ideal situation, when talking about application installation, is when the distribution has a shop application that includes *Snap* and *Flatpak* as well, giving the user maximum flexibility. On the other hand, we appreciate native GUI tools that are specific to a given format, too.

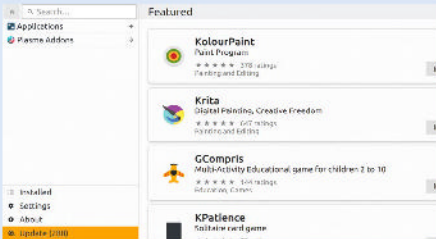
If the shop application of a distribution doesn't support all the packaging formats that you're interested in, you are stuck with having to check all the channels separately yourself. Multiple installation channels also give you a second chance when an application isn't working properly.

Traditional package managers 7/10

There is a great deal of variety among the repositories and front-end tools that each distribution offers. Most of the mainstream distros now offer a store-type application as well as a component-level front-end, but not all of them do.

By and large, the official package repositories opt for a cautious approach when it comes to following the latest releases. On a system component level, this offers maximum stability but can mean you miss out on the latest features. When it comes to apps, even the biggest distributions are often months out of date.

Because the whole system would have to be updated, it might be impossible to add the latest version of an app, and this puts system package managers at a disadvantage. There are systems such as PPA (on Ubuntu) for adding up-to-date software, but they have their own issues.

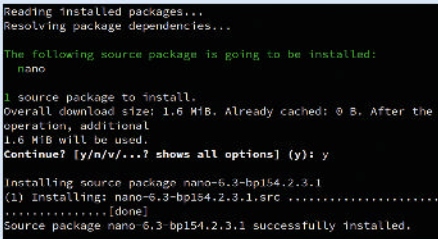


Source packages 5/10

Source packages are provided by the developer, and there isn't a central shop. However, most distros can fetch source packages from the official distro repositories. For example, on a Debian-derived distro such as Ubuntu, *apt-get source* fetches the source code for a package, and there are distributions such as Gentoo that specialise in offering source packages as the primary format.

As with traditional binary packages, if obtained from the repo, a source package often lags behind the latest release. Obtained manually, source packages can be held back by components in the Linux distro in terms of newness.

If there is a store type of app for the distro, reviews are generally for the app itself along with how well it runs on the distro when installed as a binary package. Source packages come into their own when you're willing to do some extra work.



Sandboxing and system separation

Can you protect your system from a misbehaving program?

By default, *Snap* packages have little access to the system. The package creators themselves are able to switch on certain access to resources. Furthermore, the user mode tools for managing Snaps allow refinement of these options. For example, the Snap Store allows users to switch the permissions of an application on and off to access the network, audio facilities and the home folder, among other resources.

Snap has a small problem with its sandboxing because it was created by Canonical and designed to work, first and foremost, within Ubuntu Linux. For this reason, it uses a system called *AppArmor* for its security features, meaning that distributions that use a different security system (such as SELinux) don't have all of the same security features when working with Snaps.

System package managers are at a disadvantage when it comes to separation from the underlying system as they are designed to add components to the system. There are methods for sandboxing, such as using a *chroot* environment or a container-based subsystem such as *Docker*. However, these

solutions don't work in every case, and they require quite a lot of expertise to get working. If you possess the skills to work with source packages, you could install the compiled package to an alternative directory and run it from there.

Flatpak and *AppImage* both offer sandboxing facilities similar to *Snap*, but they're not biased in favour of Ubuntu in terms of their features. The *AppImage* format has some other advantages in this area because the package runs from the archive and can even run from a read-only medium. At no point does an *AppImage* need root access to the system as it is never installed.

VERDICT

FLATPAK	8/10	APPIIMAGE	9/10
SOURCE CODE PACKAGES	5/10	SNAP	7/10
TRADITIONAL PACKAGES	4/10		

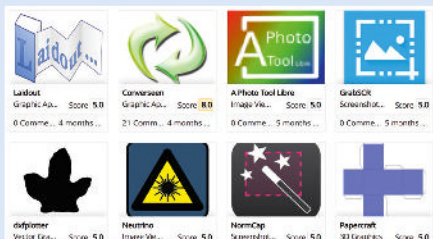
AppImage leads the way when it comes to independence from the underlying system, but *Snap* and *Flatpak* have strong features in these areas, too.

Applmage**6/10**

Applmage is often used to provide a downloadable archive that is independent of a central database. Consequently, it is popular with developers who simply want to distribute the latest version of their software via their own website.

There is an official *Applmage* web store called ApplmageHub (www.applmagehub.com), with over 1,200 apps. Reviews and ratings are offered, but we weren't blown away with the level of engagement. In addition, the ratings averaged on the low side, which could be regarded as a positive or negative.

We weren't particularly impressed by how up to date packages on ApplmageHub are, so it's usually a good idea to check the official website of that app, as developers sometimes offer a more up-to-date *Applmage* package. If you're browsing within a specific category, make sure the Latest tab is selected.

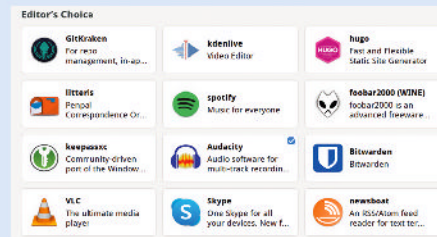
**Snap****7/10**

Many distros include the Snap Store app. It's searchable, and each entry includes the all-important user reviews and ratings.

The fact that the Snap Store relies on a central server operated by Canonical is a common criticism. It's not impossible for third parties to start their own Snap repositories but uptake has been minimal, and if a developer wants to publish their software on Snap Store, they have to be OK with Canonical controlling it.

The best situation for dealing with Snaps is when the store app of a distro includes Snaps alongside other packaging systems, so you can compare versions, and we wish this were the case with more distros. As it stands, you might have to check different repositories separately.

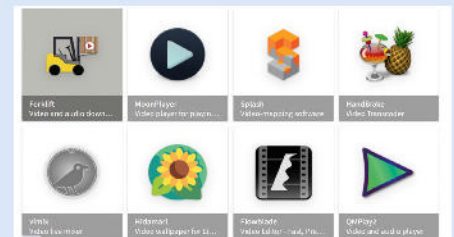
The Snap Store, in its application or web incarnations, isn't particularly well suited to chase down individual components, as its focus is application installation.

**Flatpak****6/10**

Flatpak uses an online shop called Flathub (<https://flathub.org>). The site has a bare-bones interface, and we couldn't find any ratings or reviews. Usefully, it gives you the command-line sequence needed to install a given package. In some cases, *Snap* packages were more up to date, and in others, the Flathub was ahead, emphasising the benefit of checking both.

Some shop apps, such as *Gnome Software*, include Flatpaks in their listings, but this means any ratings and comments are probably for the app itself rather than the *Flatpak* version. Bearing this in mind, we'll leave it for you to decide if the lack of an official *Flatpak* shop app is a problem.

Flathub is the largest *Flatpak* repository, but setting up an independent *Flatpak* repository is fairly simple. The packages are hosted on a normal web server. KDE, Gnome and Fedora are examples of projects with their own Flatpak repository.



System integration

We want an installed app to fit in with the rest of the user experience.

Traditional package managers have an advantage for integration with the underlying system as installing components into the system is their raison d'être. This means that an application usually inherits the desktop settings of the system and is available from the application launcher of the desktop environment. We would expect system components installed in this way to be available to other installed software.

Installing from a source package can cause integration problems. Although this method adds the component or application to the system, often the package manager and the desktop environment of the system are unaware of its presence. However, typically, the application runs in much the same way as it would have done from using a traditional package manager though.

Applmage doesn't score highly in this section as it goes to quite some effort to make sure that it's not too tightly integrated with the system. Sometimes applications running from an Applmage fail to pick up on the desktop settings of the host

machine. This might be a minor annoyance, but it can be a bigger problem if the font size ends up being tiny, for example. There are workarounds, and it's often possible to pass environment variables to the application that you wish to launch, but that feels at odds with the general easy to use nature of Applmages.

Snap and *Flatpak* offer something close to a best-of-both-worlds compromise in this area. Typically, if you install an application using either system, it fits in with the desktop environment, but components are often not universally available, unless the particular package is designed to work that way.

VERDICT

FLATPAK	8/10	APPMAGE	6/10
SOURCE CODE PACKAGES	6/10	SNAP	8/10
TRADITIONAL PACKAGES	9/10		

Flatpak and *Snap* offer good integration, but nothing can beat traditional packaging systems in that area.

Component installation

How easy is it to install components?

If you install from the distro repository using a native packaging system, that system should be able to retrieve the dependencies for you. If you are manually installing from an archive such as an RPM or DEB, you may have to go on a hunt to track down the needed dependencies, and this can be complicated. If you want to add something outside of the repository system (such as a newer version of a component), you can run into brick walls when it can't be done.

Things go up a notch in difficulty when it comes to upgrading individual components by using a source code package, because you have to locate the development dependencies in addition to those needed to run the component. Because there are fewer safeguards in place, the chances of making your system nonstandard or even outright breaking the system are higher than with the other approaches.

Snap started life as a way of adding components to Ubuntu server systems. For this reason, it's often a reasonable choice when adding components, including drivers. Although *Flatpak* is more widely known as a method for adding applications, it's perfectly capable of installing system components. For example, OpenSUSE adds proprietary graphics drivers using *Flatpak*.

```
killermike@localhost:~> flatpak info org.freedesktop.Platform
ID: org.freedesktop.Platform.GL.nvidia-470-161-03
Ref: runtime/org.freedesktop.Platform.GL.nvidia-470
Arch: x86_64
Branch: 1.4
Origin: flathub
Collection: org.flathub.Stable
Installation: system
Installed: 1.9 MB

Commit: 191497526e6e68028e116581986be3f261b1f17bb1b199
Parent: 7821f21c0e90bbd00ce651118c59da0be4879d8d3f8258
Subject: Add Vulkan 525.47.04 (bb436063)
Date: 2022-12-20 05:17:01 +0000
killermike@localhost:~>
```

Using "flatpak info" shows us that OpenSUSE is using Flatpak to manage the Nvidia graphics driver. As ever, there's complexity to the package naming conventions.

AppImage isn't generally used in a system component installation role, but it is worth noting that, just like *Snap* and *Flatpak*, the libraries can be included in the *AppImage*. This means that it is often a useful approach for installing an application that needs a more up-to-date dependency than the distribution repository can offer.

VERDICT

FLATPAK	8/10	APPIMAGE	5/10
SOURCE CODE PACKAGES	6/10	SNAP	8/10
TRADITIONAL PACKAGES	8/10		

For the mundane, everyday stuff, the system package manager is usually the way to go, but *Snap* and *Flatpak* have some advantages.

Other pros and cons

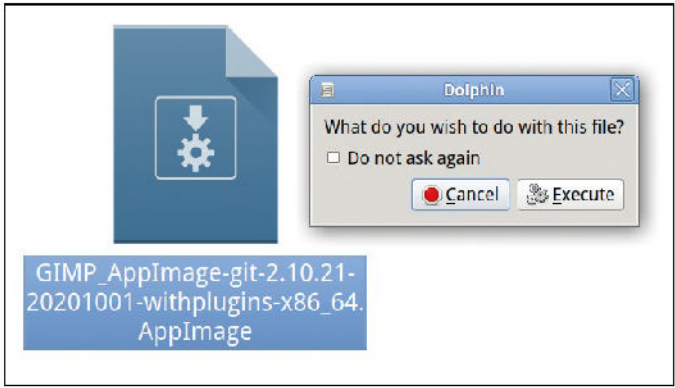
Any features that stand out?

Some formats include the dependencies within the archive. Apart from the advantages of portability, this means the developers (or packagers) can make their own choices about the included versions of libraries, which can lead to a more stable and performant version of an application.

Independence from the system leads to larger package sizes. In the case of *Flatpak*, *AppImage* and *Snap*, you could have a package where the app itself was the smaller part of the archive. Traditional package managers take a different approach, and in some cases, only a small amount of storage is required to add an app to the system when using a tool such as *apt* on a Debian-derived distro. Having said this, it's worth considering that apps tend not to take up a lot of disk space.

It's useful to be able to install multiple versions of a single app, and traditional package managers struggle with parallel installations (and the associated support libraries), but *Flatpak* and *AppImage* make it easy to do. *Snap* can also do it, although the documentation currently lists the feature as experimental and it has to be enabled from the command line. With *AppImage*, there are no command-line switches to mess with as you simply download and run the archive, and retain the older version, too.

Getting a source code package compiled can be a lot of work, and as with manual installation of binary packages such as RPM, there's a limit to how recent a version of an app you can install, as



An *AppImage* archive contains all needed dependencies, and it's never installed to the system. Perfect for trying out the latest version of an app without committing.

the support libraries often have to be upgraded. Next-generation package managers such as *AppImage*, *Flatpak* and *Snap* are less limited by this, which can make choosing a distro that's less likely to use up-to-date libraries more workable.

Traditional package managers, *Flatpak* and *Snap* can list all of the packages on a system and update them. This is a weakness of *AppImage* as there is no central database of installed apps on your machine. The same can be said for source code packages.

VERDICT

FLATPAK	8/10	APPIMAGE	6/10
SOURCE CODE PACKAGES	5/10	SNAP	8/10
TRADITIONAL PACKAGES	5/10		

Traditional package managers are the foundation that holds Linux together, but the alternatives are in the lead with the exciting features.

The Verdict

Packaging formats

We've chosen *Flatpak* as the overall winner. Obviously, most Linux distributions require a system-level, traditional package manager, but when it comes to installing the latest version of an application, a system such as *Flatpak* makes a lot of sense.

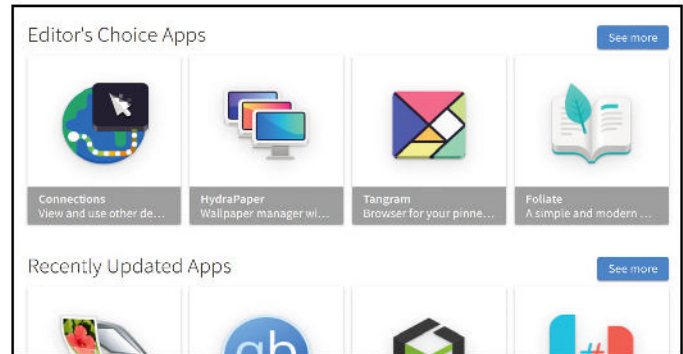
It has good command-line tools for listing and updating the applications that it is managing on a system, and we liked the way that *Flatpak* handles the installation of system-level components, such as graphics card drivers, on a system like OpenSUSE. Parallel installation of apps is a really useful feature. It lacks an official GUI front-end, which is a point against it, but the shop-type application on many distros lists *Flatpak* packages along those of the native package format.

Canonical's *Snap* has many of the same aims as *Flatpak*. In many cases, the two systems are so similar that it's difficult to choose which would be the better choice when installing software. Some elements of *Snap* are tied to Canonical, and some parts of the whole ecosystem are not open source. For this reason, we found that downloadable *Flatpak* files were a more common sight than the equivalent *Snap* archives.

Traditional package managers are the mainstay of Linux distributions. Packages avoid duplication by utilising the libraries and other resources that are already on the system. Sometimes, a desired package is out of date when using the official distro repository, leaving you stuck if getting it would involve a system-wide upgrade that isn't possible. Manually installing packages in formats like RPM often needs a close match between the target system and the one that the package was created on.

AppImage is particularly strong in a particular use case: installing the latest version of an application without changing the system that you're installing on. Feature-wise, although it handles this case extremely well, it is lacking in other areas, such as maintaining a system-wide database.

There will always be technically minded users who are willing to install an application from the source code packages. This can lead to a journey that is fraught with hidden dangers and perplexing difficulties, but it is sometimes the only way of getting the absolutely latest bleeding-edge package on to the system. If anything, the next-generation package managers that we've looked at have largely negated the necessity of doing this these days, though.



1st **Flatpak** 9/10

Web: <https://flatpak.org> **Licence:** LGPL-2.1-or-later

Version: 1.12.5

Open source and decentralised. Our first choice for adding up to date apps.

2nd **Snap** 8/10

Web: <https://snapcraft.io> **Licence:** GNU General Public v3.0

Version: 2.57.6

Up-to-date packages, excellent management tools and some tasty features.

3rd **Traditional package managers** 7/10

Web: N/A **Licence:** Various

Version: N/A

The mainstay and foundation that keeps a typical distribution on its feet.

4th **AppImage** 7/10

Web: <https://appimage.org> **Licence:** MIT

Version: Archive dependent

The simplest way of adding up-to-date applications to a distribution.

5th **Source code packages** 5/10

Web: N/A **Licence:** Package dependent

Version: Package dependent

Largely a last resort these days, but still worth considering.

» ALSO CONSIDER

Every standard Linux distro includes tools to manage packages. The RPM and DEB format packages are the most common, but there are others. Arch Linux uses a package manager called *Pacman* and PKG files, and Puppy Linux uses its own PUP files that are designed for single-click installation.

Favourites might come down to personal preference, but they all accomplish much the same thing. The exception would be packaging systems that use source code that is

then compiled on the target machine, such as the *Portage* system used on the Gentoo distribution. A setup like this has some potential for greater performance once the package is installed, but it leads to greatly increased installation times.

Containerisation formats such as *Docker* are a bit like packages because they can be used to install applications independently of the system. Most of what's available in this form tends towards server-type applications, though. **LXF**

FREE YOUR PHONE!



Lesser-spotted *Jonni Bidwell* climbs to the top of a 5G mast to extol the benefits of running Linux on your phone.

The mobile device world has become duopolised by two big tech outfits, one fruity and the other Borg-ish. Google's Android runs on an array of mobile hardware, has a Linux kernel, and the core Android Open Source Project (AOSP) is freely licensed. Devices can generally be unlocked and custom apps can be sideloaded. Compare it with Apple's walled-garden, locked-down hardware approach in iOS, and you can see why free software advocates might prefer to side with the Android camp. But

this argument is specious, because much of Android proper is proprietary.

The Google apps and most of the popular Android apps are all proprietary, and many rely on the fiendish Google Play Services library to do their work (or display ads). Those apps all collect data and tie it to your Google account. Despite new privacy controls, permissions and supposedly easier-to-understand license agreements, it's difficult to comprehend what is being shared and how it's used. This has led to an orthogonal argument, that Apple (a fashion brand that makes

hardware) cares more about protecting privacy than Google (an ad company).

We're not here to settle arguments, though. We're here to show you another way. How to take back your privacy. How to keep old hardware no longer supported by Android ticking along. To that end, we'll look at installing LineageOS, the most popular AOSP derivative. Next, we'll explore running 'pure' Linux in the form of UBports and PostmarketOS. If you'd rather not install a whole new OS, there are plenty of less risky measures you can take. Read on...

The issue with Android

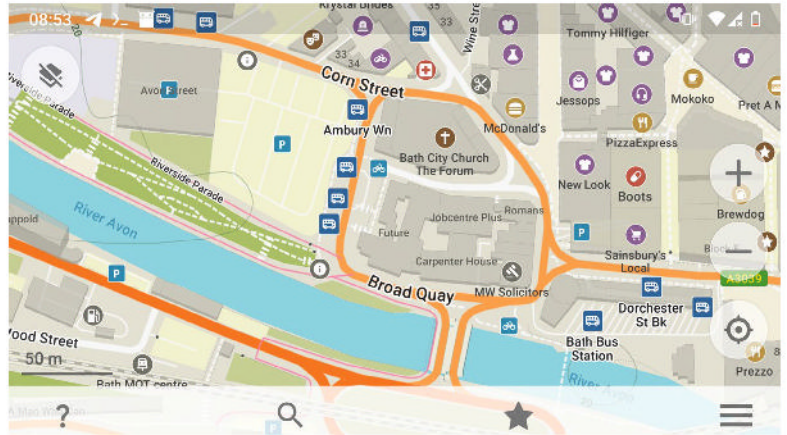
Android is a hodge-podge of free and proprietary components.

Our perennial Escape Google features are always fun, but escaping the Borg on mobile deserves a treatment all of its own. From the moment you put your Google Account credentials into the setup screen, your phone/tablet/TV stick starts feeding data to Google's silos. Search history, GPS location history, call logs, contacts and much more are by default shared with the company. Its PR outlets boldly claim that it doesn't sell this data, but it does use it to sell highly targeted ads. As you'll have noticed.

Pause for a second to consider the resolution of the profile that all this data provides. Then pause for another second to consider the second-level data obtained by cross-referencing it with billions of other users. By now you might be feeling ill at ease. But fear not, because we're here to help you free your phone and restore your peace of mind.

Replacing the OS on your primary phone is not something that should be done lightly – unlocking the bootloader voids your warranty, and installing the wrong build may brick your device – so we recommend going through the process on a different device first. Even if you don't want to go so far as to replace your device's OS, you can still reclaim some freedom by avoiding the proprietary Google apps (collectively known as G-Apps, such as *Maps*, *Gmail*, the characterless OK Google voice assistant) and in particular the Play Store. You might want to avoid other proprietary apps and services, too, and the F-Droid store can help you do just that.

Android 13 certification requires 2GB of RAM (and that's for the cut-down Android Go edition). Technically, this shouldn't stop anyone using a AOSP13-based ROM on a 1GB phone, but the experience may not be pleasant. Devices with 8GB of storage have long



struggled to cope with more than a couple of apps installed, though this can be alleviated by moving those apps to an SD card. Older, low-end hardware (such as our decade-old Moto G) tends not to be officially supported by LineageOS, but it's difficult to find a device for which some enthusiast on the XDA developer forums isn't maintaining a custom ROM.

The current trend is for OEMs to support flagship devices for about four years, with other devices often receiving much less attention. For example, Sony's XZ1 launched in 2017 with Android 8 and received an OTA (Over The Air) upgrade to Android 9 in early 2019. When Sony chose not to update that hardware to Android 10, some users were so aggrieved that they launched a petition, which didn't really accomplish anything (*I'm shocked!-ED*). Thanks to projects that build off AOSP – LineageOS, Replicant – users of such neglected devices are able to enjoy new Android features, enhancements and importantly security fixes.

Ditch the proprietary Android apps and run the entirely free LineageOS. Here's Organic Maps, showing Future Towers and surrounds.

» MOBILE LANDFILL AND HOW TO STOP IT

To ship a phone running Android, the manufacturer must submit it (together with its drivers and Android customisations) to a certification process by Google. Certified devices receive regular security updates from Google, as well as updates from manufacturers, if they design to provide them.

Carrier-locked phones additionally require the phone company's support to

receive an Android upgrade, and that upgrade may also be blocked pending contributions from the manufacturers of the device's components.

Since manufacturers often don't bother porting new versions of Android to their older hardware (and when they do, users don't install them), there is considerable fragmentation in the Android ecosystem. Some very capable hardware is stuck

running old software. Not only does this contribute to the Android landfill problem, but it's a very real security concern. Android 8.1 Oreo launched in 2017 and received its last security update in October 2021, yet according to Statcounter figures (for January 2023), over 7% of Android devices run Oreo, and about 5% run something even older.

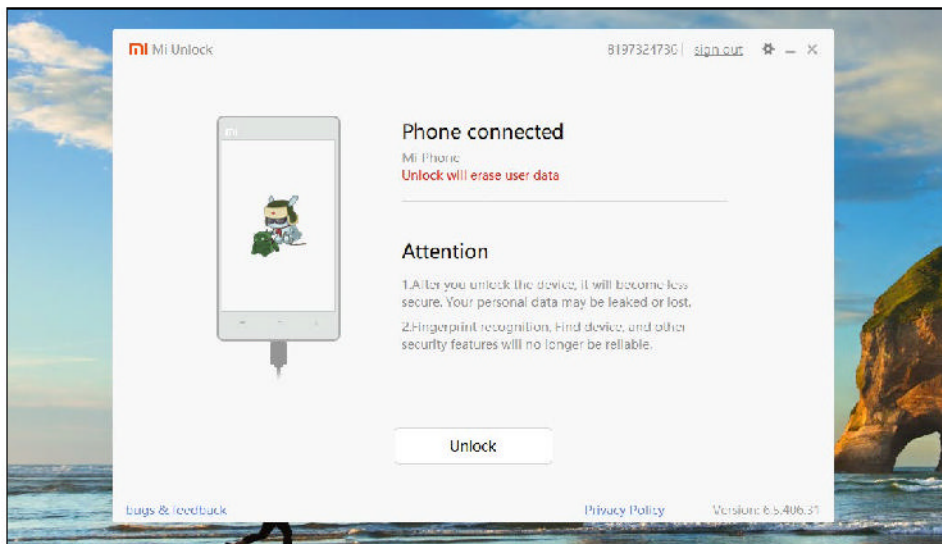
But things are much better than they used to be. In 2017,

Google's Project Treble sought to standardise a "vendor interface" in Android 8.0. This decoupled hardware-specific code from the rest of the OS, enabling updates to be carried out independently of the vendor. Two years later, Android 10 was announced, which included Project Mainline, an effort that allowed core parts of the OS to be updated via the Play Store, rather than waiting for a full OS upgrade.



Jump to LineageOS

Enjoy fully featured Android without all the proprietary bits.



Xiaomi's unlock tool makes you jump through all sorts of hoops. We used a burner SIM and throwaway email to register.

The Android Open Source Project is where manufacturers start when they want to port Android to a new device. Manufacturers add in the drivers (and other hardware-specific code), as well as perhaps custom UI overlays. If this all passes Google's muster (see box on previous page), the device can be shipped with Android pre-installed. Those wanting to compile custom ROMs for that device may find that the manufacturer has open sourced the code they added in (namely device trees, hardware libraries and things that end up in the vendor partitions), but this is rarely the case. Even if they provide something, drivers for the GPU, cameras, GPS and the baseband modem itself will more than likely only exist as proprietary blobs. Those are no use for compiling

AOSP, so custom ROM developers must find another way.

LineageOS is the most popular AOSP fork, having emerged from the ashes of the almost-as-popular CyanogenMod in 2016. LineageOS is much easier to port to other devices as the team has developed shims and layers so that vendor blobs can be used natively there. As such, its official builds offer across-the-board hardware support. So, newer hardware can enjoy the latest Android features without having to wait for manufacturers' updates. And older hardware neglected by manufacturers can stay secure.

In order to access the Play Store and the plethora of proprietary programs there, users can add a G-Apps package at install time. We strongly recommend trying life without G-Apps, but they must be flashed alongside the ROM they're being used with. If you decide post-installation that you do need them, you need to start over. There's a number of third-party G-Apps offerings available, and we've dedicated a box to them (see page 41).

Before you go rushing to install LineageOS, do be careful doing this on a device you're currently relying on. Be sure to back up any photos, contacts, messages and the like. They may already be stored in the cloud (hopefully not on a proprietary service), but do check as they won't be on your device once you unlock its bootloader. Also make sure said device is officially supported by LineageOS. You can check this at <https://wiki.lineageos.org/devices/>. Be sure to check the

» THE PERILS OF BOOTLOADER UNLOCKING

Unlocking your bootloader, in theory, opens you up to lots of attacks that compromise the device at boot and so circumvent any higher-level protections. This shouldn't be much of a concern as long as you're not sideloading Russian downloaded APKs. But malware has been found in the Play Store before, so take care. LineageOS does provide signed builds (which include verified boot) and ought to play nice with locked bootloaders, but it's all very experimental still. Samsung

devices with Snapdragon processors mostly ship in the US with perma-locked bootloaders. And many carrier-locked devices don't allow you to unlock the bootloader either.

Unlocking the bootloader wipes your device, and you might notice a serious drop in picture quality with the standard LineageOS camera app. If you try to sideload the device's native camera app, it doesn't work, thanks to DRM keys (used by the app's secret-sauce post-

processing magic) being erased. This can be worked around if you can find a port of the Google Camera app for your device. Read more at www.xda-developers.com/google-camera-port-hub/. Finally, there have been reports that banking apps, HD Netflix playback and certain games (such as *Pokémon Go*) do not work with unlocked bootloaders (or rooted devices). Note that DRM keys are not restored during a factory reset, although the bootloader lock is re-instated.



LineageOS should be your first port of call when seeking a replacement phone operating system.

exact edition of your hardware, as flashing an incorrect image will result in tears. In theory, you should always be able to boot to your phone's recovery image to remedy this, but you shouldn't count on it. You need to check your phone's bootloader can be unlocked, too.

We've gone a bit quiet on the subject of Apple since the intro (*but the Google digs have been hard 'n' fast* – Ed). That's partly because it's quite hard to unlock its bootloaders (without resorting to exploits). However, *Project Sandcastle* allows the iPhone 7 and a couple of others to run Android (or pure Linux) from a sandbox, without touching the bootloader. There's no LineageOS for iThings, either, so we won't mention them again.

Bring your phone into Lineage

We got our hands on a Xiaomi POCO X3 Pro (*hey, give that back, it's mine* – Ed), so let's have a crack at unlocking its bootloader, disposing of MIUI and getting into Lineage. A quick gander at the LineageOS devices page brings us to <https://wiki.lineageos.org/devices/vayu/>, which provides an outline of the whole process, as well as telling us the device's code name is Vayu. First we need to enable Developer Mode (in order to enable bootloader unlock). On most devices, this involves going to Settings > About Phone and repeatedly tapping on Android Version. It's similar for the X3, except here we hammer on MIUI Version.

Now we come to the bootloader unlocking. If you don't have a Xiaomi device, you're spared the joys of signing up to MIUI (tying your phone to said account), downloading its Windows-only app and waiting a week for an unlock code. We had to undergo that misfortune, but it was worth it. Some manufacturers force you through different hoops to obtain an unlock code, and some (older Samsungs, via the open source *Heimdall*) let you proceed without one. The process differs wildly between phones, so check the docs for your device rather than blindly following our instructions.

For many devices, the bootloader unlockery involves obtaining an unlock code from the manufacturer's website. To do this, we need to send (usually by a web form) a token particular to the device. Sometimes the form just asks for the IMEI number, other times it's obtained using *Android Debug Bridge (ADB)* and/or *Fastboot*. On Ubuntu (and friends), we can install the required ADB and Fastboot tool with:

```
$ sudo apt install adb fastboot
```

With the phone in Developer Mode (see above), we then go to Settings > Developer Options and activate both OEM Unlocking and USB Debugging. Now

connect the device to your Linux machine, approve the debugging prompt and run:

```
$ fastboot get_identifier_token
```

You may need to reboot your phone into Bootloader (or Fastboot) Mode for that to work. In that case, run:

```
$ adb reboot bootloader
```

Follow the manufacturer's instructions to obtain an unlock key using that token. Now reboot the device and unlock the bootloader with:

```
$ fastboot oem unlock <key from manufacturer>
```

If you don't need such a token (on Google devices, for example), the command is:

```
$ fastboot flashing unlock
```

Now the LineageOS install process becomes more uniform. The first step is to install a custom recovery environment. We show you how to install *TWRP (Team Win Recovery Project)*, but do follow the LineageOS guidance – the *Lineage Recovery* tool is recommended for some devices. Download the appropriate TWRP build for your device from <https://twrp.me/Devices/>. While you're at it, you may as well grab the LineageOS ROM by following the link on the LineageOS wiki page for your device (and optionally a G-Apps package). With your device still connected to your PC, switch to Bootloader Mode and flash TWRP with:

```
$ fastboot flash recovery twrp.x.yz-codename.img
```

Some devices let you then boot that directly, with:

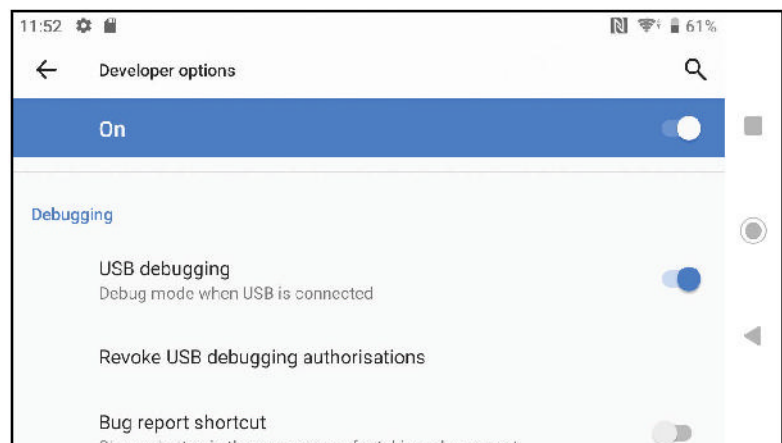
```
$ fastboot boot twrp.x.yz-codename.img
```

Others require you to manually enter the recovery environment with a combination of Power and Volume buttons. Once you've got to the TWRP menu, format the data partition by choosing Wipe > Format Data and typing **yes**. Your data is all gone, but there's more to delete. From the Wipe Menu, choose Advanced Wipe, select the Cache and System partitions, and swipe to confirm. Now we can install LineageOS. On the main menu, go to Advanced > ADB Sideload and swipe right.

Send over the LineageOS ROM you downloaded:

```
$ adb sideload lineage.x.y-...codename.zip
```

Do the same if you want to install a G-Apps package (see page 41), remembering this needs to be done before booting LineageOS. When done, choose Reboot from the main menu and enjoy LineageOS. Turn the page to find out more about your newly freed device. If Lineage is not to your liking, page 38 explains how you can run pure Linux on mobile devices instead.



Activating Developer Mode evokes memories of a sweaty Steve Ballmer chanting "Developers! Developers! Developers!"



Get to know LineageOS

We take a quick tour of the latest OS to power your mobile device.

Without doubt, LineageOS is the premier way to experience open source Android on the widest range of consumer hardware. There's a huge community of super-talented developers (most active on <https://forum.xda-developers.com>) constantly updating and building LineageOS releases new and old for specific models of devices. So, even if there's not an official release for your device, a quick DuckDuckGo will likely return someone on XDA proffering a build – with varying degrees of device support; do you *really* need the camera? – for your obscure Chinese-made mobile device. Let's take some time to look through the latest release details of LineageOS 20, but also some of the other features and advantages the project offers.

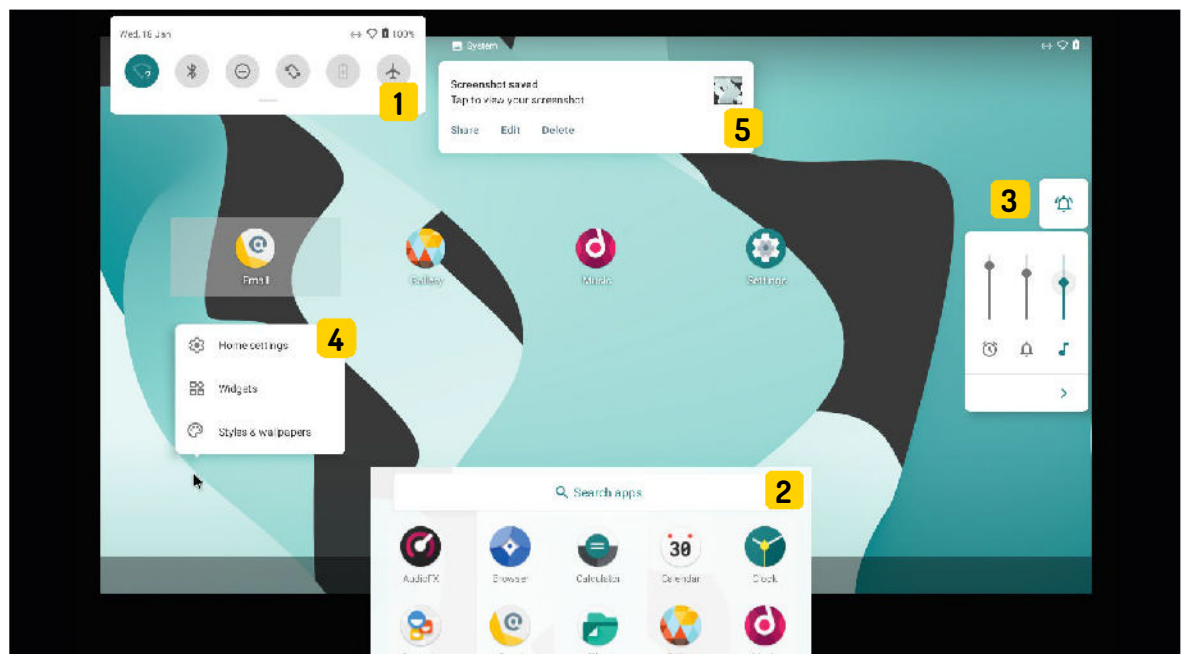
As we mentioned, LineageOS rose from the regrettable disintegration of the Cyanogenmod project in late 2016. This also explains the odd version numbering LineageOS has inherited, where Android 13 is LineageOS 20, not LineageOS 13 – that would be too

sensible. Back in the day, Cyanogenmod major release versions matched the early Android 1.x point releases; so Cyanogenmod 3/4/5/6 matched Android 1.5/1.6/2.x/2.2.x. Once Android 5.x arrived, Cyanogenmod 12.x kept pace with point and major releases.

LineageOS 20 matches Android 13, because part of this latest LineageOS release is to tweak the versioning it uses to match the new quarterly release schedule for Android. From the blog: "As Android has moved on to the quarterly maintenance release model, this release will be LineageOS 20, not 20.0 or 20.1 – though worry not, we are based on the latest and greatest Android 13 version, QPR1."

A key point of using LineageOS for your mobile device is to secure it, as manufacturers generally do a bad job of rolling out any sort of updates for devices that have gone off sale. Even though Google makes its security updates openly available, you may not get them until six months after release, if at all. LineageOS releases always bundle the latest security patches –

LineageOS desktop



1 Quick access settings
Entirely configurable, this pull-down menu offers fast access to your most-used system settings. Toggle networks, data settings, your torch and much more in just two actions.

2 App drawer
The aptly named Trebuchet launcher gives you swipe-up access to all your installed apps. With people suffering app overload these days, there's a handy search right at the top so you can filter through to the one you need.

3 Audio controls
As devices have become ever more proficient at handling media, so the requirement for finer control over that media has grown. The slide-out audio control is one step towards that, with better system, alarm and notification settings.

4 Control your home
No, not that home – your mobile device's home desktop. If you long-press on a blank area, you are able to configure how the home

looks, select your choice of wallpaper and add active widgets such as the classic clock and media controls.

5 Notifications
Modern app notifications can be a real annoyance. While well-behaved programs offer genuinely useful notifications – as seen here, taking you directly to the relevant image for editing – many are simply annoying. Long-press to gain access to settings that can mute or limit app notifications.

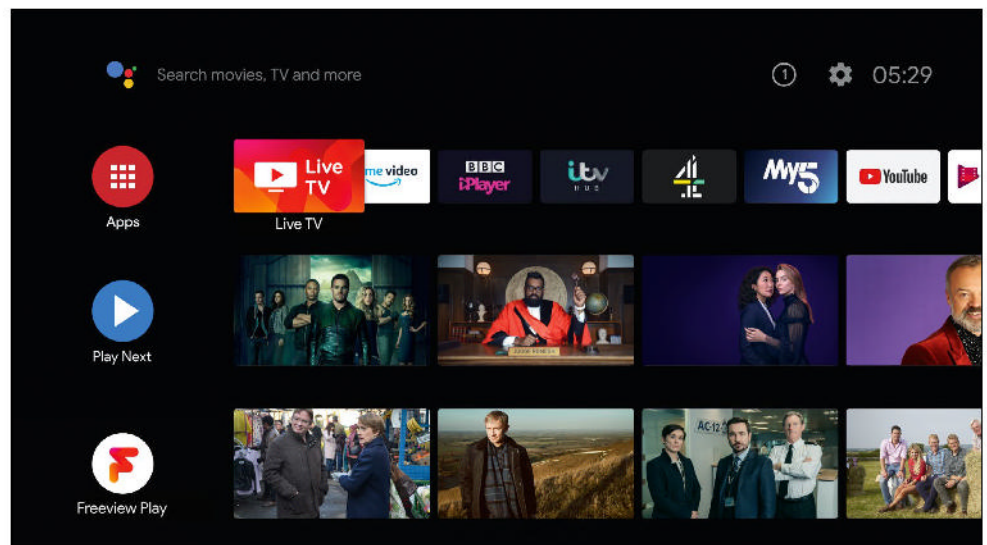
from the release notes: “April 2022 to December 2022 have been merged to LineageOS 17.1 through 20.”

There are many behind-the-scenes changes to LineageOS; one of the bigger deals is GKI support. This is a Google initiative and it stands for the Generic Kernel Image project. Long story short, the Android version of the Linux kernel is a bit of a 'mare – effectively, every device has its own version. This makes security patches tricky, LTS merges hard, upgrades harder and it's difficult to support upstream Linux. This is why Android can be stuck using kernels as old as 3.18 that require lots of out-of-tree code and backports.

The GKI (see the official site <https://bit.ly/lxf299gki>) attempts to simplify the situation by spinning out vendor SoC code to separate vendor modules with a stable Kernel Module Interface, so these modules can be independently updated. LineageOS 20 supports GKI 2.0 with Kernel 5.10, and it's already reporting that moves like this are freeing up developer time to focus on actual features.

Talking of features, one of those is a new dedicated camera app called *Aperture*, which is making its debut in LineageOS 20. The previous camera app, *Snap*, was a fork of a Qualcomm developer app that was ungainly to support, so it reverted to the very basic *Camera2* AOSP default. The LineageOS team wanted something that was a little more featureful, so a project called *CameraX* reached the point where it could be polished and included as the new *Aperture* camera app. Apparently, this is still very much a work in progress, so you can expect more features to drop over time, but even so, it offers video controls, full electronic and optical stabilisation control, and a leveller.

The integrated calendar is supplied by the Etar project (<https://github.com/Etar-Group/Etar-Calendar>). It provides an open source calendar that can work fully offline but can also use Android calendar



■ This is how Android TV can look, but probably won't on your Raspberry Pi.

sync to tie into Nextcloud, Google Calendar, Exchange and so on. It offers the usual month, week or day views.

Another welcome addition to LineageOS is a phone backup solution called *SeedVault* (<https://bit.ly/lxf299seed>). Most Android users are accustomed to the background cloud backup Google uses with Android, and *SeedVault* is an alternative way to achieve the same result. It enables Android users to store their phone data without relying on Google's proprietary cloud storage. You can decide where your phone's backup is stored, with options ranging from a USB flash drive to a remote self-hosted cloud storage alternative such as Nextcloud. *SeedVault* also offers an Auto-Restore feature – if an app is removed and later reinstalled, *SeedVault* restores its settings alongside.

Android TV support without the integrated Google ads is also on the way for supported devices. If you look at the Raspberry Pi boxout (below), there is a specific Pi build of Android TV, though just how useful this is we're not sure, as the DRM Widevine version is the minimum level, so only SD streaming is permitted by any services requiring it, if it works at all. However, as a 10-foot interface option for big screens with remote navigation baked in, it's worth keeping in mind.



» RASPBERRY PI

No one makes builds of LineageOS for x86 PCs; they'll point you in the direction of the Android-x86 project – and rightly so. If you snoop around the LineageOS FAQ (<https://wiki.lineageos.org/emulator>), you'll find detailed instructions on how to compile your own emulator that can run LineageOS rather slowly on a PC, but it's aimed at developers and requires 32GB of memory and 300GB of space for the latest 18.1+ releases.

The simplest way of trying out LineageOS – if you don't have a suitable spare phone – is to crack out your Raspberry Pi. Now, there's good news and bad news. So, the bad news is that if you want to try LineageOS 18.1 or newer, you need a Pi 4 with 2GB of memory – Google bumped the minimum specification just recently to 2GB of memory. The good news is that for 1GB Pi 3 and Pi 4 models, you can try builds of LineageOS 17.1 – this is

from the end of 2020, so remains relatively recent.

A developer called Konsta keeps a repository of non-commercial LineageOS builds for the Pi. You can find a list and what they target at <https://konstakang.com/tag/LineageOS/>.

Download the ZIP, extract the archive, check the md5 if you like, and use `dd` or, for a GUI, use www.balena.io/etcher. Write the image to a spare SD card and boot your Pi with it. Simple!



Install GNU/Linux

Touch base with Ubuntu Touch or branch out into PostmarketOS.

Early Nokia smartphones (notably the classic N900 from 2009, *see over there on the right*) shipped with a Linux-based OS called Maemo. Maemo was largely based on open source projects, but included some proprietary code. The community was key to developing it, under Nokia's stewardship, and it fuelled the vision that one day our phones could run free software without compromise. That may not have fructified yet, but there is hope. Today, long after Nokia largely abandoned Maemo (being forced by Microsoft to sell Windows Phones in 2011), there is still a community dedicated to Maemo. Namely Maemo Leste, which runs an upstream kernel and is based on Devuan (the distro that most has it in for *Systemd*).

You may not be interested in running Linux on hardware quite as old as the N900, but our FOSS-on-phones story doesn't end there. That utopian vision was rekindled with the Ubuntu Touch project, which promised to bring Ubuntu to mobile devices, smart TVs and pretty much anything MC Hammer could touch. This was serious news in 2011, and Canonical even attempted a massive crowdfund for the Ubuntu Edge phone in 2013. But, alas, it all came to naught. In 2017, Ubuntu Touch was shelved (alongside Ubuntu's characteristic Unity desktop) as interest dwindled and the company sought to make itself more appealing to investors (and fans of the Gnome desktop).

All was not lost, though. Such was the interest in Ubuntu Touch that the project was forked into a community effort under the aegis of UBports (see



Ubuntu Touch has a novel interface. A long swipe left brings up this handy overview.

<https://ubports.com>). Since taking the reigns in 2017, the project has expanded support to nearly 100 devices (OK, some are more supported than others) and issued 24 OTA upgrades. To see whether your device is supported, visit <https://devices.ubuntu-touch.io/>. If it is, and you want to give Ubuntu Touch a whirl, continue reading. You need to unlock your bootloader for this, so please read all the warnings on the previous pages about what that entails.

There's a friendly installer for UBports that works for most devices. Some legacy hardware, such as our BQ Aquaris 4.5 (one of the first Ubuntu Touch devices), requires a different approach, so do study that device wiki carefully. Hardware that requires legacy installation (such as our BQ here) does, if it's running Android, need to be flashed back to the legacy version of Ubuntu Touch OS, and this requires the use of the (even more) legacy *SP Flash Tool*. From there, the UBports installer can be used as below (except that you need to specify the device manually).

If you're running Ubuntu, the easiest way to get the installer is as a Snap, otherwise it's available as a DEB and an AppImage. Follow the links in the device's wiki, then launch the app. It asks you whether you want to install some Udev rules. We found these weren't necessary and our device was detected right away, but your mileage may vary. They can be added in later from the Tools menu if you need. Under the hood, this communication happens over *adb*, which we met on the previous pages. It's important that your phone is running the version of Android that matches the *Halium*

» HALIUM, LIBHYBRIS AND FRIENDS

Several OSes, including non-legacy UBports images, are based on a remarkable bit of software called *Halium*. *Halium* provides a unified hardware abstraction layer (HAL, geddit?) on top of the vendor-provided Android kernel. Coupled with some *Systemd* gymnastics and Libhybris, which communicates directly with binary blob bits of Android, this allows developers to build new flavours of mobile Linux from a common base. *Halium* thus aims to reduce the oft-decried fragmentation in the mobile Linux ecosystem. It works its magic via an LXC container running a minimal version of Android. Libhybris then marshals communication with the Android libraries (built with Google's *Bionic*) and Glibc libraries on the underlying Linux system.

There has been great progress developing all the gubbins peculiar to a mobile OS. The *oFono* telephony stack originated as a joint effort by Nokia and Intel in 2009. It's been used in Ubuntu Touch since the start and is also used by Lomiri, the continuation of Unity 8. Most of the UIs you can install on PMOS use *ModemManager* to deal with calls, text and data traffic. As for the user-facing aspects, we've got *Chatty* for texts, the *Matchbox* on-screen keyboard and plenty of other apps (see https://wiki.postmarketos.org/wiki/Applications_by_category). If you want to run a tiling window manager on your phone, there's *SXMO* (*Simple X Mobile*, see <https://sxmo.org>).



The classic Nokia N900 – not enough to save the company.



(if indeed you're using a *Halium* build – see box) version of the build you're trying to flash. You can check this on the device wiki. Check the documentation (<https://docs.ubports.com>) once you are up and running.

First past the PostmarketOS

If you want to run full-blown Linux on your phone, that is with all the programs and UIs you're (presumably) running on your desktop PC (and ideally running an upstream kernel), look no further than PostmarketOS (PmOS). As the name suggests, the project's primary goal is to grant end-of-life devices a new lease of life. Unless you have a PinePhone or Librem 5, there's no official hardware support, though there are dozens of community-supported models (including the venerable N900 and older Samsung Galaxies). Alternatively, you can run it in a virtual machine (VM). Start by installing PmOS's bootstrapping utility, which isn't yet in the Ubuntu repos (but is there on Arch, Debian and Fedora). Then we prepare our workspace for building the image:

```
$ git clone https://git.sr.ht/~postmarketos/pmbootstrap
$ cd pmbootstrap
$ ./pmbootstrap.py init
```

You're asked for a working directory and then a release channel. The defaults should be fine, so hit Return to accept them. Next you're asked for a target device vendor, code name and kernel – happily, the defaults build the required QEMU image. Then you're asked for a username and desktop environment. We're going with the rather slick `plasma-mobile`; try `sxsway` for the lightweight SXSO interface running on the Sway Wayland compositor, or anything that takes your fancy. Next you're asked to confirm build options. Be aware that building requires lots of free space in the working directory, and that artefacts are all cached (for faster rebuilding). So, you may want to tame some settings here. The last questions we're sure you can figure out for yourself.

The next command prepares the chroots where the system will build (so as not to interfere with host packages). We'll add in `NetworkManager`, but networking can also be set up manually (in the VM itself with PmOS's `setup-interfaces` command) if you want a lighter system:

```
$ pmbootstrap install --add=networkmanager
```

You're asked for a password – note that this doubles as a PIN for some interfaces (such as Plasma Mobile), so it's best if it only contains numbers at this stage. Don't worry – you can change this inside the system later. So, let's fire up our VM with:

```
$ pmbootstrap qemu
```

By default, the image is minimally sized, leaving only a little space for extra programs. Use the `-image-size` parameter to change this. Our Plasma Mobile target builds a 4GB image, so we don't need this. Note there's no sound in the VM; in theory, it should work if you add `--audio=pa`, for example, to the previous command, but this did not work for us. Never mind, you should be presented with an odd-shaped window and a login screen. Enter your PIN, then swipe (drag) left to open the Applications menu. Here you'll find the *Angelfish* browser, *Elisa* music player and a few things that don't work (such as *Dialer* and *Megapixels*, a camera utility).

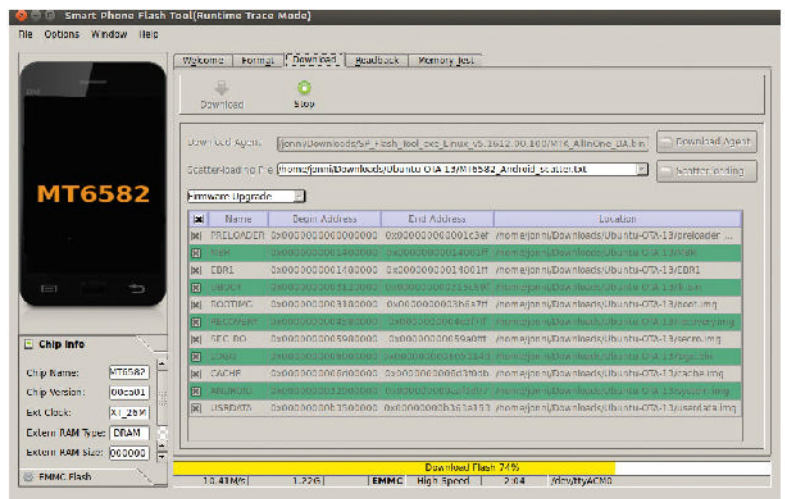
Do keep an eye on the size of the working directory by running:

```
$ sudo du -sh ~/.local/var/pmbootstrap
```

After getting to this stage, ours was already nearly 14GB. It's safe to delete everything in here when you're done playing with virtual phones. But so that the chroots are all unmounted first, you need to run:

```
$ ./pmbootstrap.py shutdown
```

The SP Flash Tool effuses a sort of esprit de '95, but for ancient phones.



Open source apps

Get into the F-Droid store and find fun and functional tools (that don't send all your data Borg-wards).



If you're now proudly running a Google-free phone OS, you'll probably be wanting to get some free (as in Mel Gibson shouting "Freedom!") apps. Even if you're still running Android and using Play Store apps, the F-Droid store is well worth checking out. It's easy to install – just visit <https://f-droid.org> on a PC and scan the QR code with a laptop. This downloads the APK (the Android package format), and when you try to install it, you'll see a warning saying the browser is not authorised to install unknown apps. This is A Good Thing. Ordinarily, we'd never tell you to download and install APKs this way, but this an exception. You can check the PGP signatures if you are concerned. Once you're happy, you can grant your browser (called *Browser* on LineageOS, and *Chrome* on Android) permission on the next screen and install.

Before we look at F-Droid, let's be sensible and revoke the browser's permission to install software. Go

to Settings > Privacy Protection > Special Permissions > Install Unknown Apps, then scroll down to your browser and deactivate it. Now we can explore and enjoy F-Droid. On first glance, it may not appear as exciting as the Play Store, or maybe that's just because it lacks brightly coloured ads and apps screaming for your cash. The first thing you should do is refresh F-Droid's cache, otherwise you'll see an error when you try to install your first app. Scroll down from the top and you'll see an 'Updating repositories' banner.

Everything on F-Droid is compiled from open source code by the F-Droid team, but it still may contain features that impinge on your privacy. Free apps may still access proprietary services, where you may potentially be tracked. *Fennec* (the mobile edition of *Firefox*), for example, sends some telemetry data back to Mozilla. Its normal build process also involves some blobs, which are removed from the F-Droid version. F-Droid

clearly labels such characteristics as Anti-Features, though in a lot of cases they're probably benign, it's just there's no way of knowing. So, if you place a premium on privacy, avoid the Anti-Features.

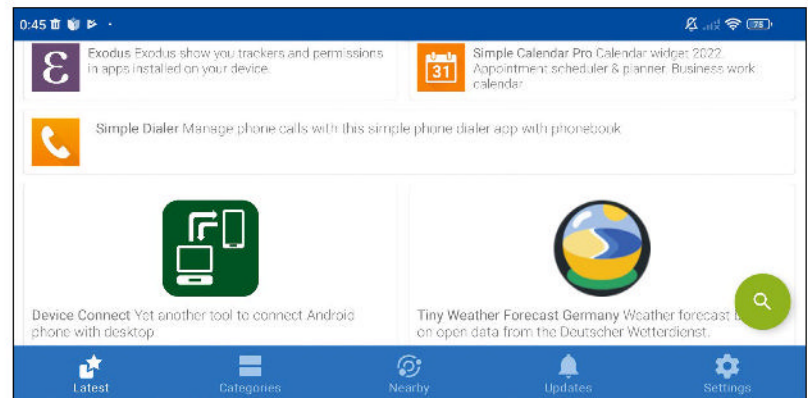
For a *Google Maps* replacement, look no further than *Organic Maps Offline*. It uses data from OpenStreetMap to draw pretty maps and give sound directions. Don't worry, you can use it online, too (you're prompted to download a local map to save bandwidth). In many cases, particularly cycling and walking paths, *Organic Maps* does better than *Borg Maps*. If you like to keep track of your runs, bike rides and joyrides, then a combination of *OpenTracks* and *OSM Dashboard* will provide at least the rudiments of your proprietary geotrackers. *OpenTracks* captures GPS data as you move around and *OSM Dashboard* plots that data on a map for you.

Amazingly, we've got this far without mention of any kind of mobile command-line interface. Let's remedy this by introducing *Termux*. This lets you incant Bash, telecommunicate over SSH or even compile code, as you would from your desktop terminal emulator. *Termux* even modifies the on-screen keyboard to provide the extra keys so vital to hackers (cursors, Ctrl). It includes a package manager, so getting started with SSH (for example) is as simple as:

```
$ pkg install openssh
```

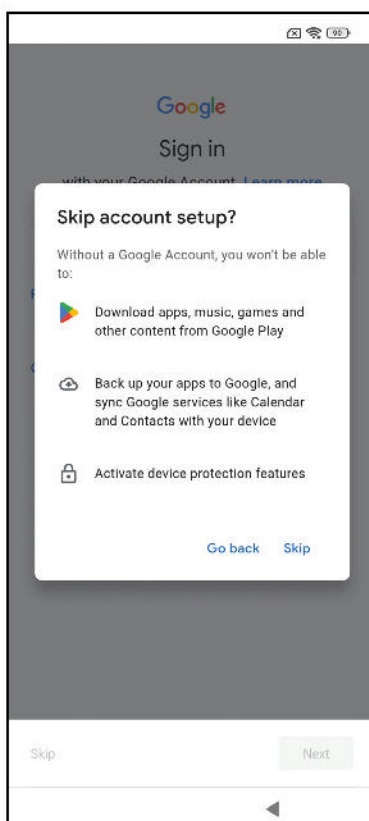
The `pkg` command is, in fact, a wrapper around Debian's `Apt`, with some tailoring to account for running on an Android base. This brings some limitations. You can't, for example, use `Apt` to install Debian or Ubuntu packages. Find out more at <https://wiki.termux.com/wiki>.

If you use Nextcloud, you'll certainly want to install the mobile app, too. There are also companion apps for its add-ons, such as Talk, Notes, News and Photos. Like many we mention on these pages, Nextcloud is also available on the Play Store, but that doesn't give you



A smorgasbord of glorious free software awaits you at the F-Droid store.

You don't need a Google Account to run Android; ignore this warning. Think of all the tracking you'll subvert.



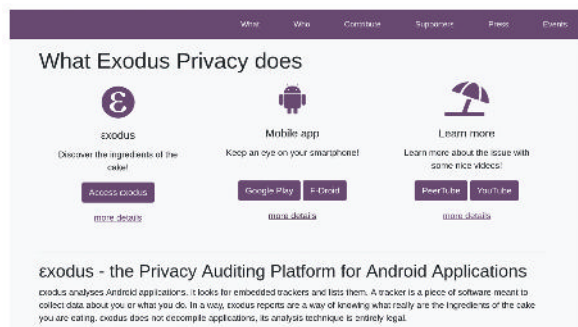
the warm and fuzzies like a build from F-Droid. Incidentally, you'll also find builds of Telegram, Signal, DuckDuckGo, Wikipedia and all kinds of other open source services you're already familiar with.

We've not explicitly stated, but it's always been possible to cut the Google cord by factory resetting your Android device. Obviously, this deletes all your data, and you won't be able to play on the Play Store, pay with Android Pay or access your contacts and so on (unless you sign back in and resync). Pre-installed apps such as *Maps*, *YouTube* and even *Google Lens* still work, albeit no longer tied to your account. Just decline the option to sign into a Google Account in the initial setup. This approach is mainly for those who don't want to nix their bootloaders. Conversely, if you really need (or want) to install Google Apps, take a look at the box below.

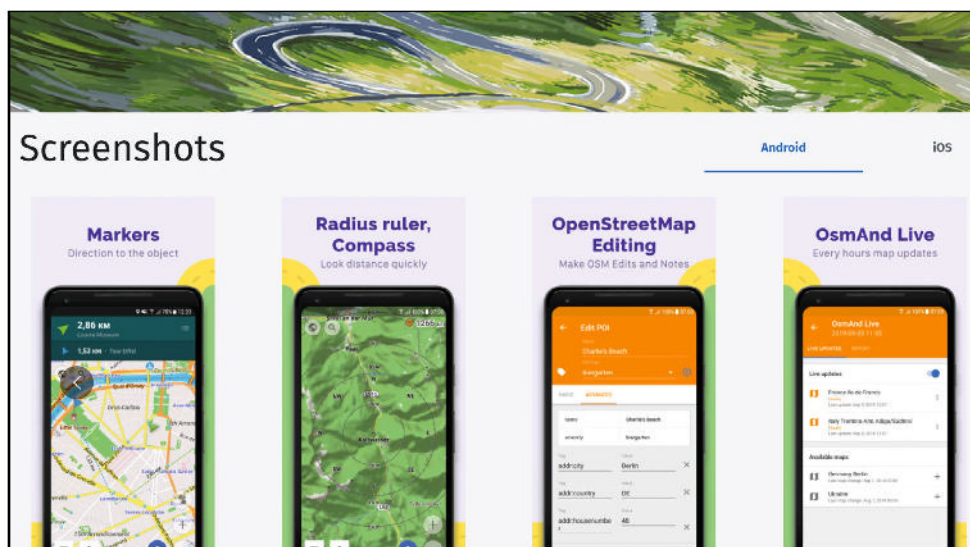
Speaking of proprietary apps, check out the *Exodus* tracker and permissions detector. If you run it on a standard proprietary-laden Android, you'll see how much everyone loves Google's Firebase Analytics. Check out the project website at <https://exodus-privacy.eu.org> for more information. If you use *Google Authenticator* as a second factor for logging into any services, consider switching to a FOSS solution. Our favourite is *Aegis*, which works anywhere the Google app would, including with Google services. Of course, switching your services away from the Google app (especially your Google Account itself) may be a painstaking process.

Notification wars

For some time, efforts have been underway to find an alternative to Google's Firebase Cloud Messaging (FCM) for push notifications. The only workaround has been for apps to maintain direct connections with their servers, which is bad for batteries (as those apps then can't be suspended). Now, thanks to the UnifiedPush (UP) protocol (see <https://unifiedpush.org>), there's a way out of this mess. Matrix and Mastodon already support UP in their apps, as does the free *FindMyDevice* app. UP decentralises matters by allowing individuals their own notification distribution



The Exodus project has an app that scans all your apps for egregious permissions and dastardly trackers.



services that interoperate with others. One such provider is Ntfy (pronounced notify, see <https://ntfy.sh>) and you can start freeing your notifications with its eponymous app.

And that's more or less all we've got space and time for. Of course, there's a lot we haven't mentioned. Perhaps most glaringly, we haven't discussed Debian's mobile efforts, dubbed Mobian. Read all about them at <https://blog.mobian.org/>. There's loads of LineageOS forks to try as well, such as the security-focused CopperheadOS (<https://copperhead.co>). We spoke of Maemo earlier, but we didn't mention that it gave rise to another mobile project, MeeGo. That one also didn't work out, but it did lead to Samsung's Tizen and Project Mer (used in Jolla's Sailfish OS, see <https://sailfishos.org>). Anyway, as ever, let us know how your mobile Linux efforts proceed. And happy hacking! 🐧

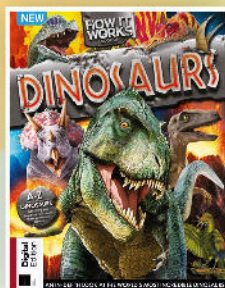
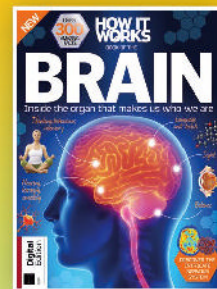
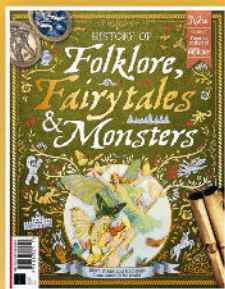
Besides Organic Maps, you might also want to check out the rather elegant, but awkwardly named, OsmAnd (<https://osmand.net>).

» DISTINCTLY CLOSED GOOGLE APPS

Some people (no matter how much we shout at the sky) want to run Google apps on LineageOS. These can't be distributed by LineageOS and have to be sideloaded at install time. New versions of Lineage recommend the **MindTheGapps** package. You'll find download links at <https://wiki.lineageos.org/gapps>. Grab the correct package for your architecture (probably arm64) and Lineage version. Sideload it from TWRP immediately after sideloading the main OS. A number of other G-Apps packages are available, some larger than others.

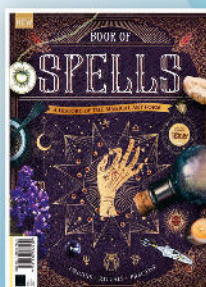
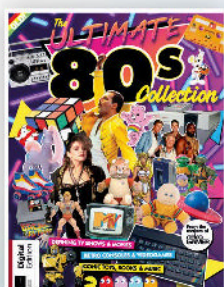
Alternatively, use the Aurora Store (on F-Droid) to access Play Store apps without a Google Account. Aurora makes it clear which apps use Google Services Framework (GSF) and whether or not they include ads. If you want to access any paid-for apps, you can sign in and do so. This might be a bit of a footgun, though – using Aurora is a violation of Google's Ts and Cs, so it might get your account banned.

The MicroG project aims to freely re-implement the services provided by Google Play Services (and more). This allows apps that depend on them to be run outside of the Google ecosystem. However, for MicroG to function properly, it requires the underlying OS to support signature spoofing, which LineageOS does not. See <https://github.com/microrg/GmsCore/wiki/Signature-Spoofing> for a list of OSes that do. Signature spoofing allows MicroG to mimic Google services, so its use is contentious.



DISCOVER OUR GREAT BOOKAZINES

From travel and history to gaming and photography, you're certain to find something you're passionate about

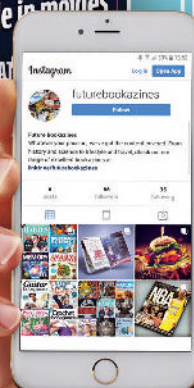


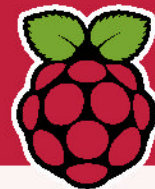
Follow us on Instagram  @futurebookazines



www.magazinesdirect.com

Magazines, back issues & bookazines.





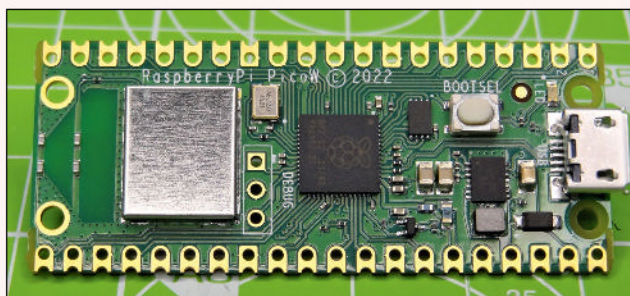
Millions and millions of RP2040s!

How to turn 500 tasty wafers into 10 million Pis!
Is everything food related around here?

In a recent Tom's Hardware interview, Eben Upton was asked, "Why is there no shortage of RP2040-based products?" Upton's answer – "We took some big risks" – led to the revelation that Raspberry Pi purchased 500 wafers in 2021.

The offer and decision to purchase 500 wafers came through within a day, Upton said. Per wafer, there can be between 21,000 and 22,000 RP2040 dies with a die size of roughly 2mm square. Upton then compared the "big chip world", where dies are 30mm square, which would gain 2,000 cores per wafer.

Upton then did the maths and from 500 wafers, each yielding 21,000 dies, there are around 10 million RP2040 chips. This



Good things do come in small packages!

'stockpile' of chips from 2021 is what many of us keen Pico users are currently consuming, be it in the form of Raspberry Pi Pico, Pico W or third-party boards. Upton then talked about what is "effectively an infinite supply [of RP2040]" based upon how many dies can be created per wafer. This is a refreshing statement, given how global supply chains have been hit by the pandemic.

More Pis in store

Unfortunately, we're talking bricks and mortar, not silicon.

According to Eben Upton, India has a massive Raspberry Pi community and it could be a viable location for future official Pi stores. These would be the first ones not based in the UK. The original store opened in February 2019, located in the heart of Cambridge – the Raspberry Pi store is found on the first floor of the Grand Arcade shopping mall.

Upton hinted at a possible second Raspberry Pi store in the UK, with Leeds looking to be a candidate. Right now, an extended pop-up store looks likely, giving



You can always p-p-p-pick up a Pi!

Raspberry Pi time to understand "what the next chapter" looks like.

Raspberry Pi has had other pop-up stores around the UK. The first was on London's Oxford Street and since then the store has moved to Gateshead and Edinburgh. For 2023, the store is back on tour and will be travelling south to Southampton, then up to Birmingham, Leeds and back to Gateshead.



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

» SAY CHEESE!

The Raspberry Pi camera was the first official add-on for our favourite single-board computer. Way back in 2013, the original 5MP model was released and over the years we have seen the megapixels increase, and even a model with interchangeable lenses. What has remained the same is a fixed focus. Though that's not entirely true – v2 could be hacked with a little scalpel work for manual focus. Now in early 2023, we are greeted with the Camera Module 3. Coming in two versions – visible light (standard) and NOIR (No IR, used for low light) – and with two lens models (75° and a wide 120°), Camera Module 3 is impressive. All of the cameras come with autofocus, giving sharp images between 10cm and infinity. Another feature of the visible light lenses is HDR, high dynamic range, which uses a series of stacked images with varying exposures to produce a single, often stunning, final image.

A good camera is nothing without good controls and we have two ways to control the camera. The Python method is via Picamera2, an updated, in-house library that gives us full control of the camera. Exposure, focus, even special effects are all at your fingertips. If Python isn't your thing, fear not as libcamera offers nearly all of the same features and it can be dropped into a Bash script.

The Raspberry Pi Camera Module 3 is available from approved Raspberry Pi resellers from around £25 for the 75° lens, while the 120° model comes in at £35. Read our full review on page 45.

Arduino 2.0.3

Les Pounder loves tinkering with hardware, so when Arduino revealed its latest IDE, he had an excuse to tinker some more.

IN BRIEF

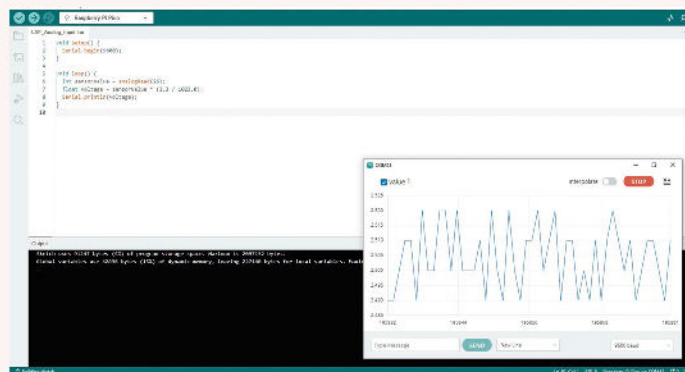
The Arduino IDE, used by makers and hackers for over a decade, has received a fresh coat of paint. But under the hood, someone has souped-up the engine.

The venerable Arduino is the genius of the current generation of maker boards. Sure, there were PIC chips before it, but Arduino democratised access to cheap hardware. That hardware was great, but the software lagged behind. Last year, Arduino released a beta of Arduino IDE 2.0 and it has now been officially released. So, we see the Arduino IDE modernised with a good-looking interface, but how does it handle?

Installation is a breeze. Right now, there are only installers for Linux, Windows and Mac (including Apple's silicon). Raspberry Pi and other Arm-based boards are left out in the cold; not for long, we hope (the older 1.8 series IDE still works with Arm-based machines). After installation is where the Arduino 2.0.3 IDE takes shape. The IDE downloads and updates the installed boards definitions and software libraries, while we expend little effort. New updates for the IDE itself are also handled in-app. No more downloading and installing another version of the IDE, then deleting the old.

Connecting a microcontroller board is now simpler. The port connection menu, just above the main coding area, shows all the available ports, and if an official board is connected, it is identified. Our elderly Arduino Uno was detected and set up in seconds. What about other boards? We connected a Raspberry Pi Pico and used the port menu to detect the port. This prompted an additional dialog box to appear, asking us to confirm what the board was. With that done, the IDE downloaded the latest board definitions and we were set for coding. A quick flash of the blink script confirmed that our Pico was running Arduino code. You can use C/C++ directly with the Pi Pico, but the Arduino IDE makes it far easier to write something akin to the optimised language.

The user interface has seen plenty of polish. We have six new menus to the left for collecting our Sketches (Arduino projects) and a link to the Boards Manager (to install new boards). We also have quick links to Software



The serial plotter is great for visualising analogue voltages, such as potentiometers.

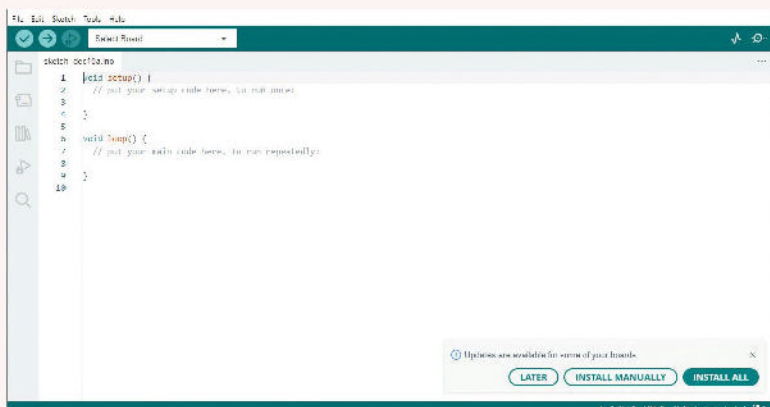
Library Management, Debug and Search. The new Serial Monitor is an exceptionally nice way to visualise the analogue data input from a potentiometer. If you prefer your IDEs dark, there is a new dark theme, which will save tired eyes working on code.

Underneath the user interface we see that the IDE has matured, and now offers code autocompletion – incredibly useful for quickly creating code. Keywords, variables and functions autocomplete as you need them. The IDE is based upon the Eclipse Theia framework and the interface is written using TypeScript. The back-end is written in Golang. All of these tweaks result in a faster compilation time, increasing productivity and enjoyment.

We've spent a considerable amount of time with the new Arduino IDE and it is leaps and bounds better than its predecessor. Not that there was anything wrong with the old version, it just felt old. We could happily work with it but now we have 2.0, it would leave us wanting more.

Just as Arduino changed the maker world all those years ago, it has firmly stamped its place as the de facto standard for makers with this new IDE. It feels grown-up and ready for professional use. The user interface is polished and the workflow has been improved, but the core still remains as it has been for many years. A great experience and one that we are very happy to repeat. 

Under the hood the Arduino IDE has seen massive improvements for board management and software libraries.



VERDICT

DEVELOPER: Arduino
WEB: <http://arduino.cc>
LICENCE: AGPL-3.0 licence

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

A polished user interface and a solid foundation see the Arduino IDE ready for many more years of service.

» **Rating 9/10**

Pi Camera Module v3

The latest version of Pi's mainstream camera module has autofocus, HDR and wide angle, proclaims **Les Pounder!**

SPECS

Sensor: Sony IMX708, 7.4mm
Res: 4,608x 2,592, 11.9MP
Video: 1080p50, 720p100, 480p120
Focus: Motorised
Focal: 4.74mm
Depth: 10cm to infinity
FoV: 66h, 41v (120-degree)
F-stop: F1.8
Size: 25x24x 11.5mm

There's no new Pi model this year but here's the Camera Module 3 at £25 for standard or £35 for the wide-angle version. The new module brings more pixels, rivalling the High Quality Camera's 12MP while keeping the smaller sensor-on-a-board form factor. What's new about this tiny camera is autofocus. This is the first official Raspberry Pi camera with autofocus, although Arducam's (see **LXF289**) high-resolution camera delivered that functionality last year.

The Camera v3 updates the original's sensor-on-a-board form factor to pack a 12MP Sony IMX708 sensor and autofocus. It also comes in four versions: standard, wide angle, NOIR and NOIR wide angle.

At the time of writing, there are two main ways to control the camera. The `libcamera` library (which replaced `raspistill` and `raspivid`) and `Picamera2` (an in-house update to the long-running community-created `Picamera`). With our Raspberry Pi 4 powered off, we inserted the camera cable and locked it into place. As the camera uses the same connector as previous models, it means it can be used with the full range of Raspberry Pi boards, including the Raspberry Pi Zero 2 W via an adapter cable.

The biggest downside of the official camera modules has always been the form factor. It lacks a mechanism to lock the camera in place. This issue was corrected with the HQ Camera's screw mount, but with Camera Module v3, we see only the legacy mount points on the corners of the camera.

Our first test was via `libcamera`. We captured a series of JPEG images. The camera was placed nine inches away from the test objects and lighting was consistent between each shot. What we found in our tests was that still images were excellent, on a par with the High Quality Camera. Autofocus times for still images were quick; we could see the camera hunting for a focus target, but it was never juddery.

Recording videos was a different matter. The focus tried to secure a target, which it did quickly, but we could see the steps as the focus worked to keep up. That said, close-up focus was excellent in our videos, picking up the surface mount components of the original Raspberry Pi.

A new feature included with Camera Module 3 is HDR (high dynamic range), which increases the dynamic luminosity range of images, producing deeper darkness and brighter images. This works by capturing multiple images of the same scene, each with different exposures. These images are then combined into a single image that encompasses the entire range.

Camera Module 3's HDR images are restricted to 3MP, which `libcamera` uses to generate an image with a resolution of 2,304x1,296. We took two pictures: one with HDR, the other without. The HDR image was unmistakably brighter and more vibrant, but it did have



The tiny motorised camera means speedy autofocus.

a little noise. The non-HDR image lacked the brightness, but also had little to no noise.

Moving our tests to `Picamera2`, we noticed that autofocus wasn't very auto. We need to include an extra class import: `picam2.set_controls({"AfMode": controls.AfModeEnum.Continuous})`.

The process and quality are identical to `libcamera`. Stills were of good quality and the focus hunt was acceptable. Video recording was a little stuttery as the focus searched for a target, but again it was acceptable.

`Picamera2` also has HDR, but not directly in the module. The issue is with V4L2, the kernel interface between the camera and the Linux video system. Right now, it does not offer ready support for HDR with this camera. In order to activate HDR for `Picamera2`, we need to open a terminal and manually set the camera to use it with `v4l2-ctl --set-ctrl wide_dynamic_range=1 -d /dev/v4l-subdev0`.

We took two images with `Picamera2`, one using HDR and the other without. The results are nearly identical to `libcamera`, but dare we say that the HDR `Picamera2` image looks a little better? **LF**

VERDICT

DEVELOPER: Raspberry Pi
WEB: www.raspberrypi.org
PRICE: £25 standard (£35 wide-angle)

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

It works out of the box and it takes great pictures. Video quality is good and more than ample for projects. HDR and wide angle are key improvements.

» **Rating 8/10**

PI PICO

Pico GPIO control with JavaScript

Les Pounder strays from Python to show us that JavaScript, the language of the web, is a viable Pico alternative.



OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at <http://bigles.com>.

Think Raspberry Pi Pico, think Python, right? That is generally the case. Be it MicroPython or CircuitPython, most makers go for this easy-to-understand language. There are alternatives – C and Arduino are incredibly useful given the Pico's low power. But did you know that JavaScript is also a viable language? In this tutorial, we introduce Kaluma (<https://kalumajs.org>), a JavaScript framework for the Pi Pico. Via three quick projects, we'll demonstrate what can be done with this rather lovely language.

The install process for Kaluma is simple, but it requires a few steps before we can write code to the Raspberry Pi Pico. Our first step is to install Node.js and npm on our system. We are using a fresh install of Ubuntu 22.04. Open a terminal and first update the APT repositories:

```
$ sudo apt update
```

Now install Node.js, followed by the node package manager (NPM):

```
$ sudo apt install nodejs
```

```
$ sudo apt install npm
```

Using Ubuntu, install the Kaluma command line interface (CLI). This is our tool to write code to the Pico.

```
$ npm install @kaluma/cli
```

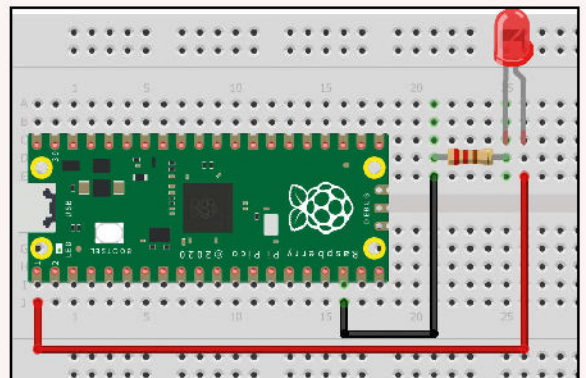
While holding the BOOTSEL button, connect your Raspberry Pi Pico to your computer. A new drive, **RPI-RP2**, appears. Open a web browser and visit <https://kalumajs.org/download>. From there click on Download .UF2. Now copy the downloaded UF2 file to the **RPI-RP2** drive. This writes the Kaluma JavaScript firmware to the Pico's onboard storage.

Our first project is a simple blink sketch, the Hello World of physical computing. For this we need to connect the long leg of an LED to GPIO 0 of the Pico, then the short leg to any GND pin on the Pico via a 220 Ohm (red-red-brown-gold) resistor. See the diagram **Blink.png** in the download for this project for details. Open your text editor and create a new file called **blink.js**. Save this to your home directory. The first line of code is a variable called **led**, which we set to use GPIO 0. Note that this variable uses the **const** keyword, which means it cannot be redefined:

```
const led = 0;
```

Next, we set the pin, **led**, to be an output:

```
pinMode(led, OUTPUT);
```



The LED blink project is a great way to test that the code and the hardware can talk to one another.

Using the **setInterval** function, we toggle the LED on and off every second (1,000ms) printing **TOGGLE** to the shell:

```
setInterval(() => {
  digitalToggle(led);
  console.log("TOGGLE");
}, 1000);
```

Save the file in your home directory as **blink.js**. Open a terminal and use the Kaluma CLI to write the code to the Pico. Press Enter to send the code:

```
$ kaluma flash blink.js
```

Kaluma detects the Pico running the new firmware, then uploads the code. In a few moments we see the LED flashing every second to prove everything works.

Getting input

We can easily control an LED in code, and now is the time to introduce an input, a push button. This toggles the LED on and off using just a few more lines of JavaScript. The button is wired to GPIO 1 and GND. Pressing the button changes the state of GPIO 1 from HIGH (ON) to LOW (OFF) and triggers the LED to toggle on/off. See **Button.png** in the download for this project.

Open a new text editor session and create a blank file called **button.js**. Save this to your home directory.

The first line of code is a **require**. Think of it as similar to **import** in Python. We use this to load a module that handles button presses:

```
var Button = require('button').Button;
```

YOU NEED

- > **Pi Pico**
- > **PIR obstacle sensor**
- > **Breadboard**
- > **5x M to M jumper wires**
- > **Buzzer**
- > **Code:** <https://github.com/lesp/LXF-299-JavaScript-on-Raspberry-Pi-Pico/archive/refs/heads/main.zip>

Next we create a new variable, **btn**, and use that to create a connection between our code and the button connected to GPIO 1.

```
var btn = new Button(1);
```

We now reuse the LED code to set the LED GPIO pin to 0, and to make it an output:

```
const led = 0;
```

```
pinMode(led, OUTPUT);
```

Next is a function that looks for a button press. When this happens, a message, **LED Toggle**, is printed to the console (shell), then the LED is toggled on/off.

```
btn.on('click', () => {
  console.log('LED Toggle');
  digitalToggle(led);
});
```

Save the code as **button.js** and use the Kaluma CLI to flash the code to the Pi Pico via the terminal:

```
$ kaluma flash button.js
```

When the code has finished flashing, press the push button to toggle the LED on/off.

Simple input and output

Let's apply our knowledge to a project that detects movement and sounds an alarm. We are using a PIR obstacle sensor, commonly used on robots. Why are we not using a generic? Because they are very sensitive and prone to setting off while debugging the code. That said, you can swap the sensors, if you wish.

The wiring (see **Alarm.png** also top-right) for this is more involved. Our buzzer is connected to GND and GPIO 12. The sensor connects to VBUS (+VCC), GND and the sensor output is connected to GPIO 28.

In a text editor, create a file, **alarm.js**. The first two lines are variables. One is a **const** for the sensor that tells code where to find the PIR and buzzer GPIO pins:

```
const pir = 28;
```

```
var pin = 12;
```

The PIR pin is then set to be an input, so we need to read the pin state to see if the alarm has triggered:

```
pinMode(pir, INPUT);
```

Using the **setInterval** method we call the contained code every 100ms. The 100ms timing is the second parameter to be passed to the method, the first being the code to make the alarm sound:

```
setInterval(function () {
```

Next we create a variable, **motion**, and in there we store the current state of the PIR pin. If there is no movement, the state is 1. If there is movement, it is 0. This is printed to the console for debug purposes.

```
let motion = digitalRead(pir);
console.log(digitalRead(pir));
```

If the sensor has detected movement, the state of the PIR pin is 0 (LOW), which triggers code to run:

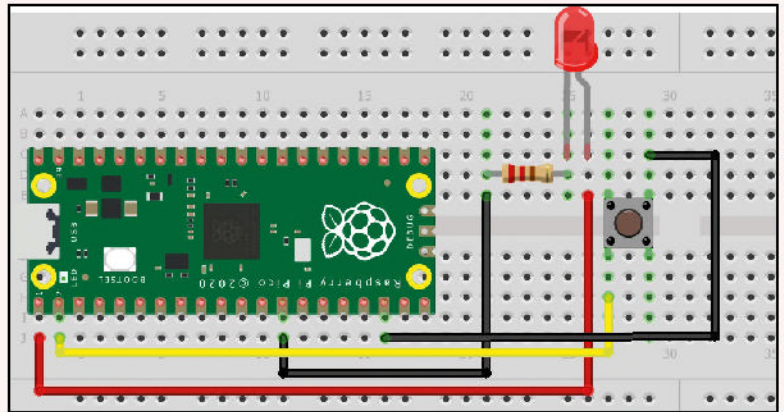
```
if (motion === 0) {
```

Create a **for** loop, starting from 0 and iterating 30 times. Each time it iterates, 1 is added to the value of **i**.

```
for (let i = 0; i < 30; i++) {
```

Using **tone**, a means to use PWM (pulse width modulation) to produce notes using the buzzer, we play C4 on the buzzer before pausing for 100ms. This pause holds the note long enough for us to hear:

```
tone(pin, 262); // C4
```



```
delay(100);
```

We do the same with D4, creating a two-tone alarm:

```
tone(pin, 294); // D4
```

```
delay(100);
```

The final line inside the **for** loop turns off any notes:

```
noTone(pin);
```

If the sensor is not triggered, the **else** condition is activated and **no movement** is written to the console.

```
} else {
  console.log("No movement")
}
```

The final parameter is added to the **setInterval** method, adding a 100ms delay to calling the method.

```
}, 100);
```

Save the code as **alarm.js** to your home directory. Then open a terminal and flash the code to the Pico.

```
$ kaluma flash alarm.js
```

Wave your hand in front of the sensor. To see the console output, open a terminal and install **Tio**:

```
sudo apt install tio
```

Pass **Tio** the Pico port, typically **/dev/ttyACMO**.

```
$ tio /dev/ttyACMO
```

You will see the JavaScript console scroll up the screen. To exit, press Ctrl+T then Q. **Tio** cannot be active when uploading code. 

Push buttons, sometimes called momentary switches, are an easy-to-use input that we come into contact with every day of our lives.

» JAVASCRIPT ON MICROCONTROLLERS

You have probably come across JavaScript while using the web. You may have been taught JavaScript as part of a computer science class, but this is perhaps your first time seeing it on a microcontroller.

Kaluma is a wonderful example of JavaScript on the Pi Pico, but it's not the only JavaScript for microcontrollers. Espruino (www.espruino.com) is a popular JavaScript framework for microcontrollers. It works with devices as diverse as tiny microcontrollers (Espruino Pico), a simple user interface device called Puck.js and smart watches such as Bangle.js. We have a number of these devices and the JavaScript framework is all web-based, enabling us to code apps via a browser.

The most famous example of JavaScript on microcontrollers is the micro:bit. Released in 2015, micro:bit was supported by the BBC and other companies to revive the '80s coding scene, in which the BBC Micro was a large part. The micro:bit is coded via the browser, using its own block language, or via Python. When coding in blocks, you could switch to JavaScript to understand how the language worked.

» GET YOUR Pi FILLING HERE Subscribe now at <http://bit.ly/LinuxFormat>

PI ROBOT

Add and monitor control sensors

Part Three!

Don't miss next issue, subscribe on page 16!

Making sense of the situation, **Les Pounder** shows us how to use sensors with a robot to get about town!



OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker. He blogs about hacks and makes at <http://bigles.com>.

At the end of part two, we had successfully built our robot from a kit, soldered the motors and tested them using a series of directional commands. To facilitate this, we created our own Python module to make things easier to control and abstract away some of the complexities. In this part, we move on to writing code that takes data from sensors and uses it to give our robot 'brains'.

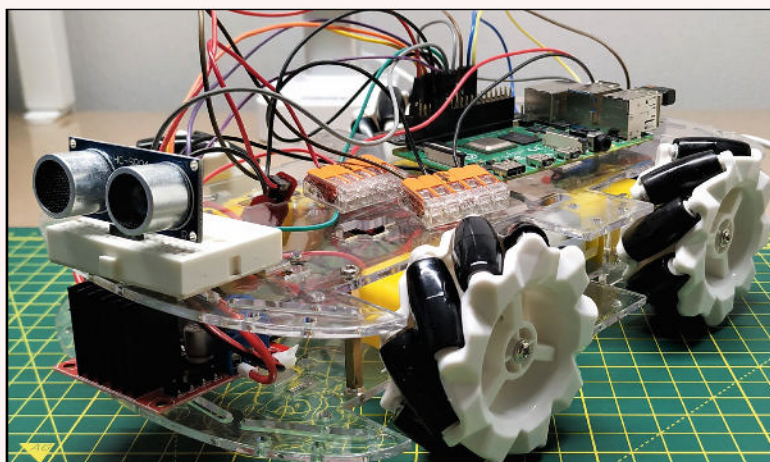
Sensors, much like our own, provide a means to see and interact with the world around us. We use our eyes to see the world. Robots can see the world around them with OpenCV (Open Computer Vision), which uses machine

learning to identify objects and people, and react accordingly. At a more basic level, robots can 'see' using ultrasonic and obstacle sensors. There are so many sensors to choose from it can bewilder even the most seasoned of makers. We've chosen to stick with the most basic forms: infrared, buttons and ultrasonic sensors. The more advanced methods of input and control will come in a future part.

This part can become quite complex – remember, we are working with multiple types of sensors. We will write the code and test each one as we go. We may use them all, or just a few. That is a decision for you. Make your robot as complex or simple as you wish – we will give you the tools and the knowledge to realise it. Start with a single sensor, learn how it works and then apply that knowledge to your build. The online resources have all of the code snippets for you to add to your code, and our **robot.py** file contains every function for every sensor we create in this tutorial.

Additional features

In part two, we created a series of motor tests (forward, backward, spins, stops) but we left one out. This isn't so much a test, rather a routine that our robot can follow when it encounters an immovable object. The routine is called **make_space** and its goal is to back the



Our robot can see, well, sort of. It uses pulses of sound to detect objects. But we can also use infrared or a good ol' button to detect an obstacle.

robot away from an object, spin and then try again. It is a function that calls the other motor functions.

Open the **robot.py** module in your favourite editor and add these steps.

Create a function called **make_space**:

```
def make_space():
```

First we tell the robot to stop, and print that to the Python shell before waiting for one second:

```
    print("stop")
    stop()
    sleep(1)
```

Then we send the robot backwards at half speed for one second:

```
    print("backward")
    backward(0.5)
    sleep(1)
```

We then stop again for a second:

```
    print("stop")
    stop()
    sleep(1)
```

We spin the robot left at half speed for 200 milliseconds. Alter this to meet your requirements:

```
    print("spin left")
    spin_left(0.5)
    sleep(0.2)
```

Finally, we stop the robot:

YOU NEED

- > The robot build from part two
- > HC-SR04 P or HC-SR04 and ultrasonic sensor
- > Obstacle/PIR sensor
- > Arcade button
- > A strip of M-to-F jumper wires
- > A breadboard
- > Code: <https://github.com/lesp/Linux-Format-Robot/archive/refs/heads/main.zip>


```
print("stop")
stop()
```

Once these steps are complete, the robot can continue on. If there is another obstacle, the routine can be called again. Save the function to **robot.py**. Later, we will use this function as a means of helping our robot escape a situation.

Plenty to sense

The world of electronics has a dizzying array of sensors. We can measure acceleration forces, magnetic fields and orientation. But the best sensors for a robot are those that give it the ability to sense the world around it. We can start basic with a bump sensor, essentially a button, then work our way to distance sensors, which use pulses of sound to measure our distance from an object. Remember that we don't need to use every sensor on our robot; choose the sensor(s) that will complement your build. At the end of this part, we will have updated our Python module to feature new functions for working with each sensor.

Bump it!

Bump sensors are the most basic of robot sensors. Really, they aren't a sensor in the truest form. Rather they are simply buttons that when pressed trigger an action to happen. Just like the buttons on a game controller. Our bump sensor can be an arcade button glued to the front of the robot chassis. As the robot touches an object, the button is pressed and the state of a GPIO pin is changed. In our case, the GPIO pin is pulled high when the button is not pressed. Pressing the button connects the button to GND, causing the GPIO pin to go low and triggering a function that stops the motors, then reverses.

The circuit for the bump sensor is a simple two-wire connection. One leg of the button connects to GPIO2, the other to GND. It doesn't matter which leg. We used a large arcade-style button for this as it is designed to take high impacts and we can easily attach it to the front of the robot. The button needs to be proud (stuck out) from the front of the chassis so the button hits the object before the robot.

Going to the code and inside the Python module, **robot.py**, that we created in part two, we now need to add a few more lines of code. We edit the first line, which imports classes from the GPIO Zero module. We already have **PWMOutputDevice** that we use to control the motors, now we add **Button** to the line.

```
from gpiozero import PWMOutputDevice, Button
```

Scroll down the code to the end – our last function was **stop()**. After this function, create a new function called **impact**:

```
def impact():
```

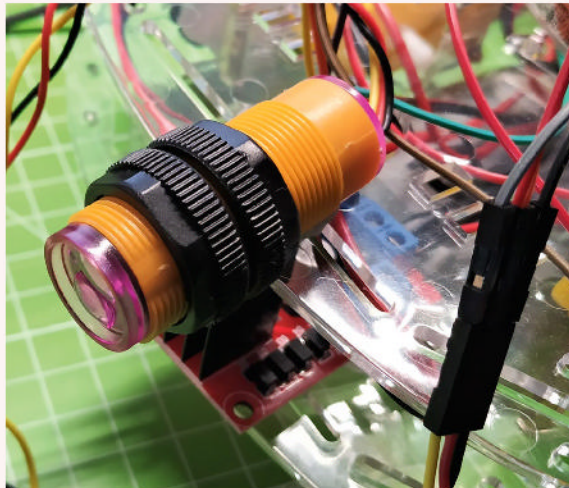
Print **IMPACT** to the Python shell. This is useful if you need to do any debugging.

```
print("IMPACT")
```

Call the **stop** function to pull an emergency stop:

```
stop()
```

We then reverse the robot at half speed for two seconds. This gives us a little space between the obstacle and the sensor:



Obstacle sensors are really PIR sensors that have a narrow and short field of vision. We can use them to detect walls, people, pets and so on, and have the robot react accordingly.

```
backward(0.5)
```

```
sleep(2)
```

Save the code to the module, **robot.py**. Now create a new file, called **sensor-test.py**. This is the test file for this input. We start by importing the **Button** class from GPIO Zero:

```
from gpiozero import Button
```

Next we import **pause** from the **signal** module. This prevents the code from exiting during our tests:

```
from signal import pause
```

```
Import our robot module.
```

```
import robot
```

Create an object, **button**, which links our hardware button to the code:

```
button = Button(2)
```

When the button is pressed, set the code to call the **impact** function from the **robot** module:

```
button.when_pressed = robot.impact
```

Finally, we need to call the **pause** function to prevent the code from exiting:

QUICK TIP

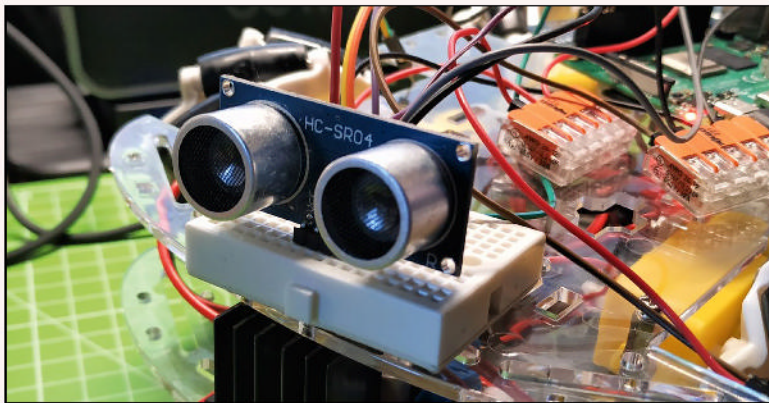
By creating a module full of functions, we have created our own, high-level interface for controlling the robot. There is no right or wrong way to control it; use the functions to create your own sequence of commands.

» SENSOR KITS

You can spend lots of money on sensors and, over the years, we have done just that. If you are starting out, it is best to pick up a kit of different sensors. These kits can be found on Amazon for £20-£50 and come with a plethora of components, including many that can be used in non-robotics projects. Tilt switches, photo-interrupter, avoidance, flame and lasers can all be yours, thanks to these handy kits. Our advice is to pick one up and use it to experiment with the sensors and your Pi. But we also suggest you pick up an HC-SR04P (or +) instead of the bundled HC-SR04. The HC-SR04 requires a potential/voltage divider, consisting of two resistors, which drops the output (echo) pin's voltage down from a damaging 5V to 3V. Never use an HC-SR04 on a Pi or Pico without a voltage divider or a logic level converter. Save the hassle and get the P/+ variant instead.

If you fancy something more sophisticated, a time-of-flight array is great for measuring distances with more accuracy. The VL53L5Cx has an 8 x 8 array that uses lasers to measure distances between 2cm and 4m. This tiny sensor can return values and basic shapes instead of distances, giving your robot a greater understanding of the world around it. It can be found for around £20 at Pimoroni.

» **DEAL WITH OUR ROBOT TEAM...** Subscribe now at <http://bit.ly/LinuxFormat>



Ultrasonic sensors are the easiest and best way to give your robot sight. The new P and + models work out of the box with the Raspberry Pi and Pico.

QUICK TIP

For Raspberry Pi and Pico-based robotics, look for HC-SR04P and HC-SR04+ ultrasonic sensors. They work out of the box with the Raspberry Pi and can be used with 3V logic (Micro:bit, Adafruit, ESP8266 and so on) and 5V logic (Arduino) boards.

`pause()`

Save and run the code. Quickly press the button to trigger the **impact** function to run. We've just created our first, basic sensor for the robot.

Passive infrared

Passive infrared (PIR) sensors are a common sensor in maker electronics. The sensors, typically BISS0001, are simple to use and relatively reliable. They are used to detect movement using infrared. Everything emits a low level of IR radiation, and this sensor is designed to detect changes, such as a person or animal moving into its field of view. When triggered, the output pin moves from high to low, giving us an event that we can use to trigger an action. Standard golf ball-shaped sensors are better used for alarm projects, but PIR obstacle sensors can be used as a crude means to detect objects and people.

We used an obstacle sensor specially made for robotics. It uses the same basic premise as a typical PIR sensor, but has a shorter and narrower field of vision. The detection distance can be tuned using a potentiometer (a screw) on the back of the sensor.

The sensor (see *top-right*) has three pins: power (VCC), out and GND. Typically these are identified by their colours (red, yellow and black, respectively). The red wire connects to 3.3V on the Pi, black to any GND pin, and yellow, the out pin, connects to GPIO4. See the diagram **Obstacle.png** in the project downloads.

With the sensor connected, we need to add a few lines of code in order to make it work. This code is added to our **robot.py** module, which contains all the

commands to make our robot work. Firstly, we need to update the GPIOZero import so that it now includes the **MotionSensor** class:

```
from gpiozero import PWMOutputDevice, Button, MotionSensor
```

Underneath where we configured our motors (rra, rrb and so on), we need to create a new object, **obs** (short for obstacle), which uses the **MotionSensor** class to configure the sensor on GPIO4:

```
obs = MotionSensor(4)
```

Scroll to the bottom of the file, past the **impact** function, and create a new function called **obstacle**:

```
def obstacle():
```

Inside the function, we use **print** to write a message to the Python shell, informing the user that the sensor is searching for an object.

```
    print("Checking for object")
```

When it 'sees' an object, it triggers the sensor to change state, and in this case it runs the **stop()** function to immediately stop the robot. We use **when.motion** because the obstacle sensor and a standard PIR sensor work in the same manner.

```
    obs.when_motion = stop
```

Save the code and open **sensor-test.py**. Clear any existing code as we are going to create a new test just for the obstacle sensor. Import the **robot** module:

```
import robot
```

Call the **obstacle** function:

```
robot.obstacle()
```

Save the file and click Run to start the test script. The function runs and prints **Checking for object**. Place your hand in front of the sensor. The LED in the sensor lights up, the Python shell prints **STOP!** and the **stop** function halts the motors. We have successfully tested our obstacle sensor.

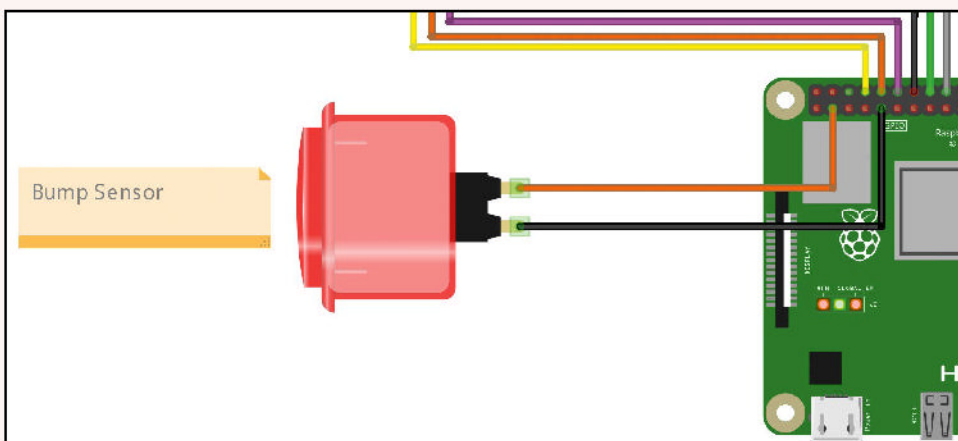
Ultrasonic the robot

You're reversing the car and hear a "beep, beep, beep". As you get nearer to the garage door, you hear "BEEP!" That is a distance sensor. It sends a ping of ultrasound at 3,440cm/s, and waits for the returning echo. The time taken for the ping to send and receive is halved, then multiplied by the speed of sound to give distance.

The common distance sensor for Arduinos and Raspberry Pi is the HC-SR04. We're using the HC-SR04P (or you can use the + model) as it is compatible with the 3V logic used by the Pi's GPIO.

The older HC-SR04 uses 5V logic, and requires a potential divider to drop the echo pin's output from 5V to 3V. In the past, we have hand-coded our own function to calculate and return the distance, but luckily GPIO Zero has a built-in function, **DistanceSensor**, that removes the maths from our code. That said, learning how to write your own function to handle this is a basic skill for most robot makers.

Going to our code and inside the Python module, **robot.py**, that we created in part two, we need to add a few more lines of code. We edit the first line that imports classes from the GPIO Zero module. We



Our bump sensor is really an arcade button, designed to take high impacts. When the button is pressed, it completes a circuit and changes the state of a GPIO pin, causing an action to happen.

already have `PWMOutputDevice` that we use to control the motors, and now we add `DistanceSensor` :
 from gpiozero import PWMOutputDevice,
 DistanceSensor

Next add an object, `sensor`, underneath where we defined the motors (`rra`, `rrb`, and so on). This specifies the GPIO pins used for the sensor, in this case GPIO27 and 17. Pin 27 is our echo pin, 17 the trigger:

```
sensor = DistanceSensor(27, 17)
```

Now we go right to the bottom of the code and create a function, `ultra` (short for ultrasonic):

```
def ultra():
```

The function first creates an object, `distance`, and in there it stores the distance measured. Note that the function returns the value in metres, so we multiply the values by 100 for centimetres. We use `round` to round the measurement to two decimal places.

```
distance = round(sensor.distance*100,2)
```

Print the distance as part of a sentence to the Python shell:

```
print('Distance to nearest object is', distance, 'cm')
```

Use a conditional test to check the distance. If it is below 50cm, call the `make_space` function that we created earlier:

```
if distance < 50:
```

```
    make_space()
```

Rise of the robots!

We've tinkered with the sensors and now it is time to put our tinkering to the test with a basic script that uses the ultrasonic sensor to measure the distance of objects ahead of our robot. If an object is less than 50cm away, the robot stops, spins and backs away before trying again. Tweak the detection distance via the `ultra` function in `robot.py`. We chose 50cm as a conservative estimate so it triggered the `make_space` function; 10-20cm would be better in the home.

Create a new file called `autonomous_test.py` and import the `robot` module:

```
import robot
```

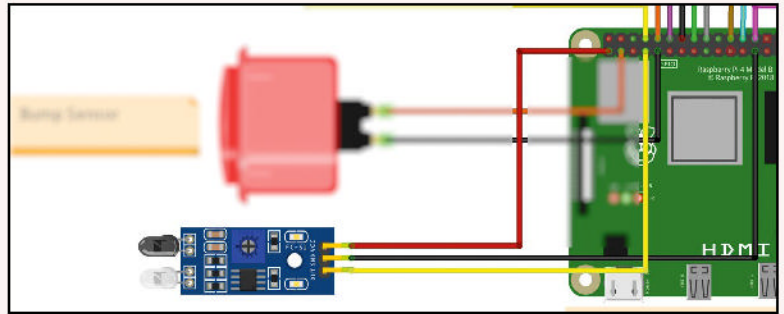
Use a `try` exception handler to run the main body of code. If this fails, an exception prevents the robot from running away:

```
try:
```

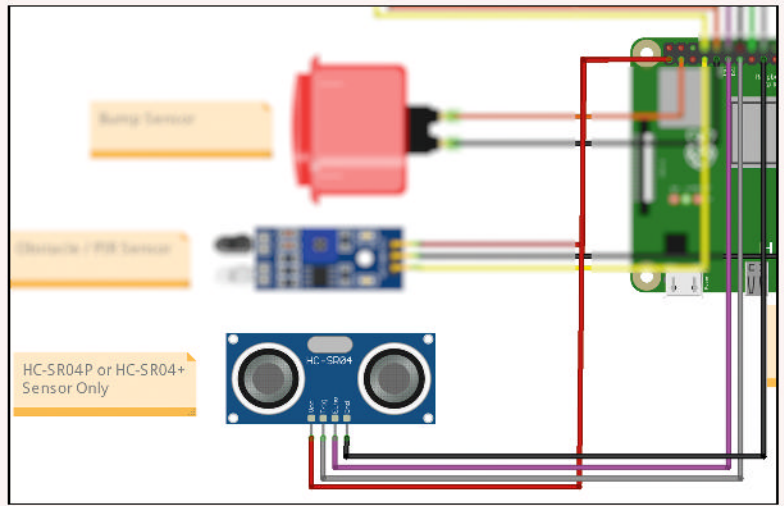
```
    Use a forever, while True loop to run the code:
```

```
while True:
```

Call `robot.ultra` to run the ultrasonic sensor, check the distance and use the conditional logic inside the function to decide whether to move forward or escape:



The obstacle sensor has three pins: VCC for power, GND and a data/output pin, which changes state when it detects a person or object.



```
robot.ultra()
```

Here is where the exception handling kicks in. Should the worst happen, we can press Ctrl+C to stop the code, print a message, stop the motors and exit:

```
except KeyboardInterrupt:
```

```
    print("STOP")
```

```
    robot.stop()
```

Save the code and click Run to test it. Lift the robot off your desk, otherwise you will be chasing it around the room. Press Ctrl+C to stop the code and robot.

Ultrasonic sensors have four pins: VCC for power, GND and a trigger pin to send a pulse of sound, plus an echo pin to listen for the returning pulse.

Where are we?

We now have a series of sensors that we can use with our robot. Our choice was the ultrasonic distance sensor, chiefly because this sensor is so easy to use and provides our robot with great long-distance sensing and obstacle avoidance. Tinker with the code, and make it your own. Remember, this is your robot. 

» MOUNTING SENSORS TO A ROBOT

Placing sensors on your robot can be tricky. Ultrasonic sensors need to face forwards, bump sensors can be placed around the chassis, line followers need to go under the chassis. But how do we connect them all to the chassis? Your approach depends on what resources you have. We can glue sensors to the chassis, using cable ties to provide strain relief to the jumper wires. Gluing Lego bricks to the chassis and using them to

make a frame is one way to secure your sensors. We can fabricate our own frames using balsa wood, or laser cut our own frames if we have access to a makerspace. The best approach is to 3D print holders or frames for our sensors, and the best thing is, many people have done so and shared their models online. Libraries such as <http://printables.com> and <http://thingiverse.com> have an overwhelming number of 3D printable

holders for our sensors, many of which don't take much time to print. Sure, you need a 3D printer, but they're coming down in price. Creality's Ender 2 Pro is a solid performer for around £150. We know, because we own one. Downloaded parts can be tweaked to suit your needs. Tinkercad is an excellent online resource to tweak 3D models. We can also use *Blender*, *OpenSCAD* and *FreeCAD* to edit and make our own parts.



No changes allowed!

Matt Holder explores immutable operating systems, containerised apps and containers to see if they describe the future of the Linux distro.

Having worked in IT departments in schools for over 10 years, painful memories exist of the agony felt by staff and students alike over the time it took to perform operating system updates. This was made many, many times worse when people began using Chromebooks, where updates were carried out seamlessly in the background and all that was required to access the new version was a reboot of the device. This is one technology used by a more modern OS, where two partitions are used to store the current OS and the upgrade, and if anything fails, the boot can revert to the older version.

In this feature, we are going to introduce Fedora Silverblue (which uses the Gnome desktop environment), which is an operating system that uses very different techniques from those you are most likely familiar with to create an operating system. Other examples of this sort of operating system are

Fedora Kinoite (using the KDE desktop environment), NixOS, which allows ultimate customisation by installing only what is configured, OpenSUSE's MicroOS, Endless OS, Ubuntu Core and Chrome OS Flex.

The first idea that we will introduce is that of immutable root partitions. What this means is that a set of files is distributed by the project and this is a rock-solid base for the operating system. Immutability also refers to the fact that parts of the filesystem are configured as read-only, with the idea being that only the operating system's developers can initiate updates to files by releasing new upgrades. Using this technique, the operating system's footprint can be relatively small, with extra apps added using various techniques, which we discuss later. Updates to the operating system are carried out by using atomic operations, which means that they are either successful or not. In this manner, it is incredibly unlikely that

updates fail and the operating system be left in an undefined state. Fedora Silverblue and Kinoite use OSTree, which can be considered a version control system for the files required to make up the OS.

Packages are added using either filesystem layers or containerised packages. The benefit here is that the base operating system no longer needs to have files edited, added or removed, which can cause stability and security issues. Tools such as *RPM-OSTree* have been created to utilise the RPM packaging system with the OSTree version-controlled filesystems. As with operating system updates, package installations are also atomic and can be rolled forwards and backwards between versions.

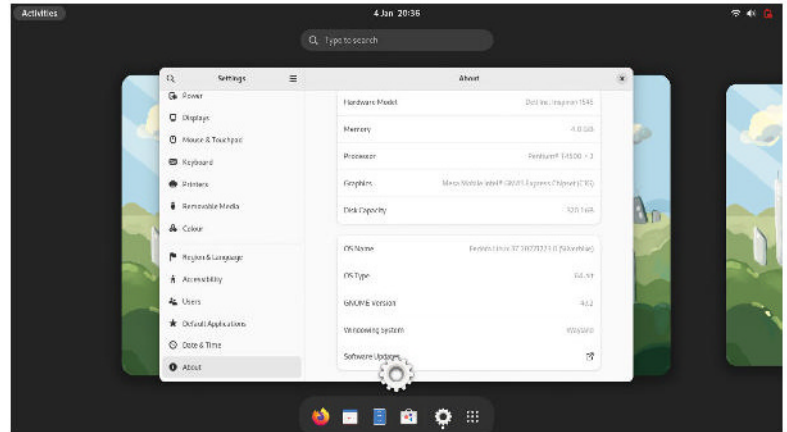
Containerised packages

Containerised packages are a very important part of this jigsaw puzzle and allow for applications to be installed as well as their dependencies, without the need to add to or alter the files on the root partition. For example, should an application need a particular Python library, this can be bundled in with the application itself. A second reason that containerised packages are so powerful is that the same version can run on many operating systems, due to the ability to use runtimes and dependency embedding.

Thirdly, as the name suggests, these apps are containerised, which means that their ability to edit the entire operating system's files can be reduced. For example, an application can be allowed access to the user's **home** directory only and given access to the USB ports. Conversely, some applications are denied access to USB.

Things have improved immensely over the last few years – for example, we remember all GUI apps, when executed from Snaps, looking dreadful because they didn't use the distribution's native theme.

Examples of containerised packaging systems (see p26) are Flatpaks, ApptImages and Snaps. The three



systems are discussed below and Snaps are the system used by Ubuntu Core, but can be made to run on other immutable operating systems as well.

Gnome 43.2 running on Fedora Silverblue.

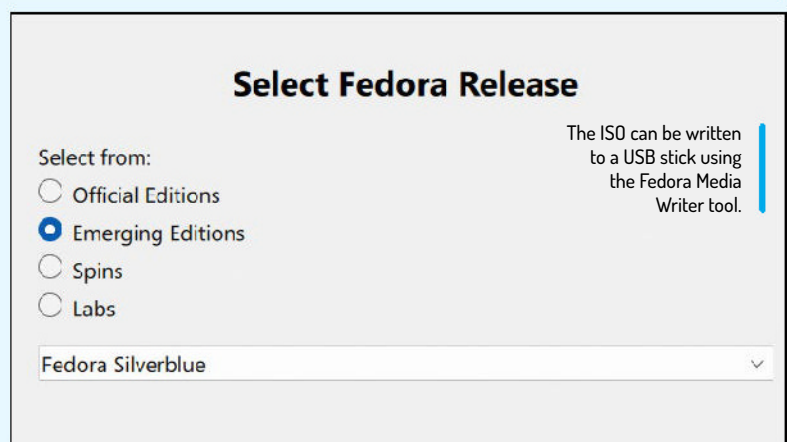
Delivered flat-packed

Flatpak is the approved format for applications to be run on Fedora Silverblue and other similar systems, such as OpenSUSE's MicroOS. To run Flatpaks, one or more runtimes are required to be installed and applications are built against a runtime. When installing an application, the required runtime is installed at the same time. If libraries or other resources are required for the application to run, these are also bundled into the file. When running an application from a Flatpak, they are contained within a sandbox and can only access their own resources and the runtime. Any further access to the filesystem or hardware must be explicitly granted. Flatpaks have a mechanism, known as Portals, which is used to allow the application to access files, hardware and other items from outside of the sandbox. Runtimes are distribution-agnostic, which means that once a Flatpak has been built, it can run anywhere that the runtime can run. Flatpaks are

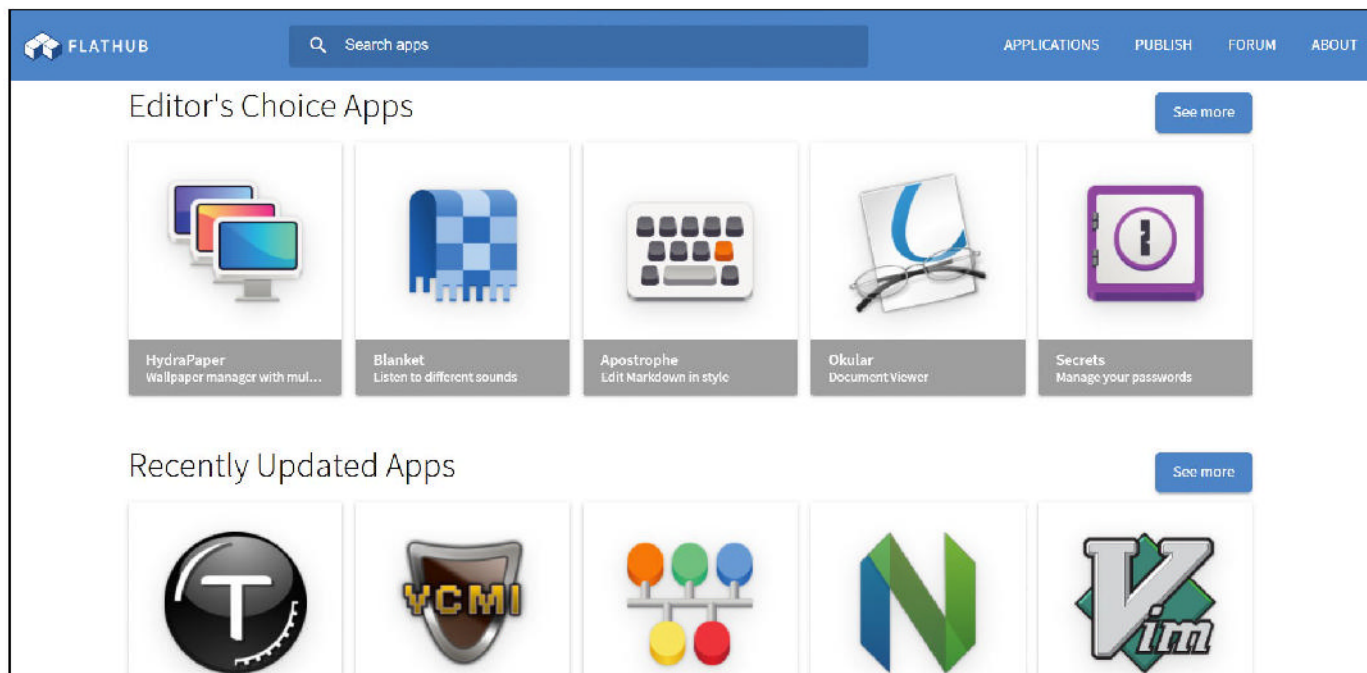
» INSTALLING FEDORA SILVERBLUE

Installing Silverblue is very similar to any other Linux operating system. Step one in the process is to download the *Fedora Media Writer* tool. This can be downloaded from <https://silverblue.fedoraproject.org/download>. Once downloaded, install the tool if needed and then run it. When the GUI opens, select Download Automatically from the first screen and click on Next. Select Emerging Editions and make sure that Fedora Silverblue is selected, before clicking on Next again. Select the latest version, correct hardware architecture and correct USB stick. Finally, select the option to delete downloaded files after the write has been completed and then click on the Write button.

The next step in the installation process is to ensure that the device you will be installing on can boot from USB. Once this is done, boot from the memory stick and follow the installer to make sure that the language, keyboard layout, location and disk to install on has been selected. The partition structure is very important for



Silverblue to function as expected, so it is wise to accept the default options and allow it to use the entire disk – use the manual partitioning option at your peril! Once completed, reboot your device and then follow the first-run tool to add a user and set the initial password.



Flathub is a repository that is used to distribute Flatpaks. The contents can be viewed from the built-in package manager and the web front-end at <https://flathub.org>.

distributed from repositories, with Flathub being the most popular. This can be added to Silverblue by using the GUI options.

Image conscious

ApplImage is a single-file format that contains all required libraries and resources for an application. The ApplImage contains a filesystem, which is mounted using the FUSE userspace filesystem mounting system. ApplImage started in 2003 and was originally named Klik. Applications packaged as ApplImages can be run on multiple Linux distributions, do not require

for some applications to open. This has been improved by changing the compression algorithm used in the SquashFS filesystem, which is used to store the files required by the application.

Installing packages

The first way of installing applications is the best method: using Flatpaks. When opening Gnome's Software store from the Silverblue desktop, all options that can be found from the discovery screen or from searching is a Flatpak. When installed, you have the security that these applications are containerised and benefit from all the points we've been discussing. The excellent audio editor *Audacity* is available from within the software centre, so search for and install this to test the system. While in the software centre, look around to see what is and is not available.

The second method is to use *RPM-ostree*. When using this method, the application and any libraries are installed as a filesystem layer and a reboot is required once this new layer has been built, so that it can be mounted in the required way. In this example, we are installing the *Fish* shell. First of all, open the terminal and enter the following command:

```
$ rpm-ostree install fish
```

Once the installation has completed, you are prompted to reboot. Once rebooted and logged in, open the terminal and enter `fish`. The *Fish* shell then opens and you can experience the benefits, such as autocompletion of command-line arguments as well as filename completion.

Keep in mind that when using this method, any installed applications are not sandboxed in the same way as when using a Flatpak. Due to this, the security and stability of the underlying operating system can be negatively affected. This method is best used as minimally as possible.

Best of both worlds

Another way to install applications that are not

WEIGHING IT UP

“The benefits of an immutable operating system clearly outweigh the disadvantages of the time taken to learn any new techniques.”

installing in the conventional sense, do not need root permissions to run and are completely portable.

Snaps were designed and initially developed by Ubuntu's parent company, Canonical, for use on the Ubuntu Phone OS. This was so that a rival existed to the way Android and iOS package apps. Snaps were then extended to support IoT applications, GUI apps and server-based options. Snaps utilise features of *Systemd* in their operation and are controlled via the *Snapt* daemon. Snaps are run in a sandbox with limited access to the parent operating system. In recent releases, Canonical has taken the decision to package and distribute *Google Chrome* and *Mozilla Firefox* as Snaps, due to the complexity of compiling web browsers for multiple operating system versions. Much criticism has been given over this due to the time taken

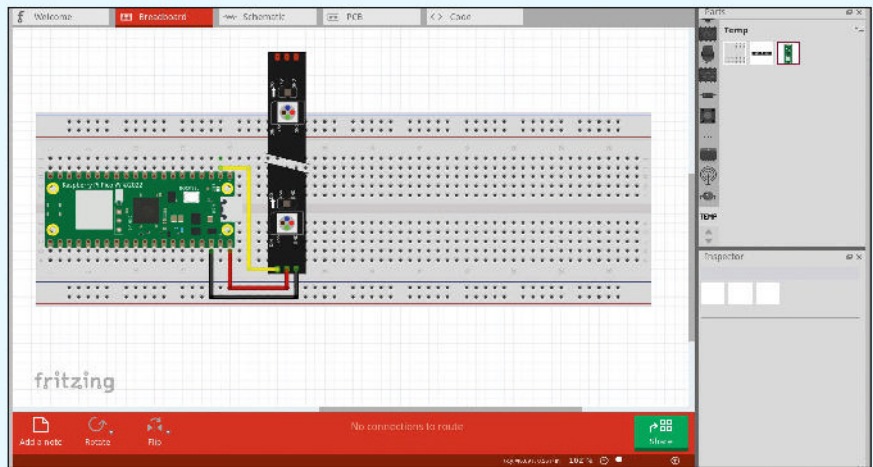
» DISTROBOX

Much like *Toolbox*, *Distrobox* is a way of running tightly integrated containers on your immutable operating system. This means the container can access the filesystem outside of the sandbox and your hardware as well. *Distrobox* can use *Podman* or *Docker* to coordinate the operation of the containers and the former would be used in this instance. Further information about *Distrobox*, including a list of URLs to use to download different distributions, can be found at <https://github.com/89luca89/distrobox>. Without using the `-i` flag and a URL, a Fedora container will be used:

```
$ rpm-ostree install distrobox
$ distrobox -i docker.io/library/
Ubuntu:22.10
$ distrobox enter ubuntu-22-10
```

Following the first command, the packages are downloaded, filesystem layer created and then on reboot *Distrobox* is installed.

Command two runs *Distrobox*, downloads the Ubuntu 22.10 image and installs it. Finally, the third command runs the container with *Podman* and opens a shell within it.



Fritzting can be seen running from the Distrobox container system.

Installing a GUI app from the container and ensure it opens as expected:

```
$ sudo apt install fritzing
$ fritzing
```

From within *Fritzting*, ensure that the file selector dialogs allow for files to be opened and saved in the **home** directory of the Silverblue installation.

available as Flatpaks is to use *Toolbox*. This is an application that allows containers to integrate neatly with the operating system. When being used, the toolbox can interact with the operating system's filesystem and resources. In the example below, we install *Inkscape* within the toolbox and demonstrate how seamless the experience can be. First of all, open your terminal and enter the following commands:

```
$ toolbox install
$ toolbox enter
$ sudo dnf install inkscape
$ inkscape
```

When these four commands are run, first of all the Fedora container is downloaded and installed. The second command starts the container and the shell then applies within the container. The third command installs *Inkscape* and the fourth starts it. Once it has opened, experiment with saving and opening files to and from your **home** directory. *Toolbox* uses *Podman* behind the scenes to manage the containers and several can be used at any one time. When installing images, any container following the Open Container Initiative (OCI) format and standards can be used. For example, to use an Ubuntu container from within Fedora Silverblue, enter the following commands:

```
$ toolbox create -i docker.io/akdev11/ubuntu-
toolbox:22.04
$ toolbox enter ubuntu-toolbox-22.04
```

Snapshots

As is often the case with open source, a project is made possible by the libraries or other projects that have been made by other people. For example, without Gnome, Fedora wouldn't be the distribution we are used to today. Another example of this is the use

of the Btrfs filesystem. Pronounced "BetterFs" or "ButterFs", this is a more modern filesystem than ext4 (extended4) and contains features such as Copy on Write and snapshots.

Snapshots are not a backup strategy on their own, but by using the `Btrfs send` and `Btrfs recv` commands, they could be, by sending the snapshot to another device. Snapshots are really handy because they could be used to save the state of a partition before and after operations. Due to copy on write, snapshots can use very little space and multiple ones can be stored of each volume or subvolume.

Given the more reliable nature of the upgrades and the smaller footprint of the operating system itself, the crucial thing to snapshot or back up is your **home** directory. Using tools such as *Toolbox* or *Distrobox*, the most important thing to make sure we can restore is our work and the configuration of anything specific we are working on. The reason for this is that containers can easily be downloaded again and the configuration reapplied to the new download.

While preparing for this article, we felt quite apprehensive about what the future would hold. This feeling dissipated quite quickly when we realised that the benefits of an immutable operating system clearly outweigh the disadvantages of the time taken to learn any new techniques and the different ways of installing software and tools.

Not all desktop environments are available to be installed on an immutable operating system, with KDE and Gnome being the main ones in use that are compatible, but as the popularity of these systems increases, more and more software will become available. Please enjoy installing Silverblue and other similar operating systems. **LXF**

Super system monitor

No sniggering at the back as **Shashank Sharma** shows you his bottom.



**OUR
EXPERT**

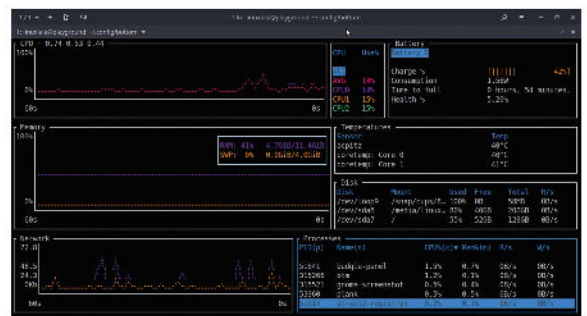
**Shashank
Sharma**
is a trial lawyer
in Delhi and an
avid Arch user.

System monitors come in all shapes and sizes. At a minimum, they provide a quick look at active processes and CPU and memory utilization. Some tools, however, go a step further and also provide information about disk and network I/O. More often than not, the system monitor also provides colourful graphs and even lets you sort or kill processes. If you think all this is only possible with graphical utilities, you're in for a surprise.

Written in Rust and released under the MIT licence, *Bottom* is a rather robust system monitor with all the features listed above, and more. You won't find it in the software repositories of most distros, the exception being Arch Linux, but the project publishes a Deb binary for Ubuntu, Debian and other derivative distros, in addition to precompiled binaries. You'll find installation instructions on the project's GitHub page for different distros such as Fedora, where *Bottom* can be installed using the Copr repository.

As *Bottom* is written in Rust, one distro-neutral option to install it is to use *Cargo*, the package manager for Rust utilities. If you don't already have Rust and *Cargo*, you can use your distro's package manager to install these – `sudo dnf install rust cargo` installs these on RPM-based distributions such as Fedora. You can similarly run `sudo apt install cargo`, which also installs the **Rustc** package, if you're running Ubuntu or Debian or one of their derivatives.

With *Cargo* installed, you can now run `cargo install bottom` to install *Bottom*. You can launch it with the



You must launch Bottom with the `btm --battery` command if you wish to enable the Battery widget.

btm command. The interface features a number of different panes, such as CPU, Memory, Processes and so on. *Bottom* refers to these as widgets. The default interface features six different widgets.

Bottom's up

The CPU widget shows a graph depicting the CPU usage. Immediately to the left of the graph you'll find a list of all the CPUs, if you're on a multicore machine, along with the percentage use for each, as well as an average. You can select individual cores, or click All or Average, and *Bottom* shows the corresponding graph, depending on your choice.

The Memory widget similarly displays the memory usage, as well as the current RAM and swap usage percentages. The Temperature widget reads the

» CONFIGURING BOTTOM

If you dislike the default colour options, *Bottom* features several different themes for you to choose from. You can switch from the default theme with the `btm --colour <theme>` command. Your choices are default-light, gruvbox, gruvbox-light, nord and nord-light.

In addition to the different command options and keybindings, you can also control *Bottom*'s behaviour by tweaking its config file. If you installed *Bottom* using *Cargo*, you can find the **bottom**.

toml config file in the `~/config/bottom` directory.

This default config file is entirely commented out, leaving you free to enable all the settings you wish to use with *Bottom*. It is another good place to start to make sense of the different command flags available for use. For instance, you can set `battery = true` in the config file if you wish to enable the Battery widget by default but don't wish to launch *Bottom* with the `btm --battery`

command. You can similarly define the colour scheme you wish to use in the **bottom.toml** config file. Additionally, you can use custom colours for the different headers such as CPU, Memory and so on, as well as the colours for the CPU, memory and network graphs.

Once you've tailored the config file to your liking, you must run *Bottom* with the `btm -C <path-to-config-file>` command so it adopts your defined parameters instead of the defaults.

relevant sensors and defaults to Celsius. You can run *Bottom* with the `btm -f` command if you'd rather have the temperature displayed in Fahrenheit.

Next is the Disk widget, which lists all the devices along with relevant information such as mount point, used space in percentage, free space, as well as read/write speeds. You have to expand the Disk widget to fully appreciate all the information on offer. This is because *Bottom* limits the amount of information on display depending on the size of the emulator window.

This is why, for optimum results, you should consider expanding the size of the terminal emulator window, or ideally maximise the window to full screen.

The Network widget shows the received (Rx) and transmitted (Tx) network traffic as a graph as well as in percent usage. Finally, the Process widget lists the currently running processes.

Although not enabled by default, *Bottom* can also show the status of the battery. The Battery widget displays the charge percent, time to battery empty/full charge and battery health.

Unlike other similar tools such as *Glances* (a system monitor written in Python that we covered in [LXF262](#)), it's possible to select individual widgets and expand them to occupy the full screen. For instance, click the Processes widget at the bottom-right of the interface. You can identify the selected widget as the colour of the boundary changes. Now press `e` and you'll notice that the Processes widget expands to full-screen mode. You can return to the main interface by pressing `e` again or hitting Escape. You can also navigate the different widgets by pressing Shift + arrow keys.

The CPU, memory and network graphs refresh every 60 seconds by default. You can use the mouse scroll wheel or the + and - keys to change this.

Process management

You can tweak *Bottom*'s appearance using different command flags. For instance, you can run *Bottom* with `-f`, `-c`, or `-k` to display temperature in Fahrenheit, Celsius or Kelvin respectively. The `--battery` flag can be used to enable the Battery widget. If you dislike the default graph style, you can run *Bottom* with the `btm -m` command to switch to a dot marker for graphs. Run the `btm -h` command for a complete list of all available

flags and a brief description, or the `btm --help` command for more details.

Bottom also supports a number of keybindings for some of the widgets, such as Processes:

Keybinding	Function
dd	Kill the selected process
t	Toggle tree view
Tab	Toggle grouping processes with same name
s	Open the sort widget
c	Sort processes by CPU usage; press again to reverse order
m	Sort processes by memory usage; press again to reverse order
n	Sort processes by name; press again to reverse order
l	Invert current sort
M	View metadata information

To test these keybindings, first select the Processes widget from the main interface and expand into it by pressing `e`. You can sort the processes based on CPU or memory usage, as well as read/write operations and so on, either using the Sort widget or directly using the different keybindings as listed in the table.

You can also kill a process with *Bottom* by first selecting it from the list in the Processes widget and then pressing `dd`.

It's also possible to search for processes by pressing `/`, which opens the Search widget.

Bottom filters results as you start typing. The Search widget defaults to case-sensitive search, as evident from the selected Case (Alt+c) entry at the bottom. Press Alt+c to perform a case-insensitive search. You can search using whole words with Alt+w.

In addition to searching for processes by name, such as `firefox` or `gedit`, you can also use *Bottom* to identify processes based on CPU or memory usage. For instance, type `cpu > 4` for a list of all processes that are taking up more than 4% of the CPU. The `cpu`

`> 4 mem > 12` search string similarly returns processes that take up more than 4% CPU and 12% of the RAM.

You can exit the search by pressing Esc. Remember to delete your search string after you're done, otherwise *Bottom* limits the processes to matching results only, if any. That is, *Bottom* doesn't revert to the complete list of processes even when you exit the Search widget. You must manually delete your search string or pattern before exiting the Search widget.

For a complete list of supported keybindings available for the different widgets, press `?` from within *Bottom* to open the Help menu. 

QUICK TIP

While we prefer the full-featured default interface, you can optionally launch *Bottom* with the `btm -b` or `btm --basic` commands. This launches *Bottom* with a basic interface that doesn't feature any graphs. You'll also have to scroll to the left or right to access the different widgets, such as Disks.

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

0s

You'll notice the Sort widget on the left and the Search widget at the bottom in this expanded view of the Processes widget.

» **ENHANCE YOUR TERMINAL-FU** Subscribe now at <http://bit.ly/LinuxFormat>

CONVERSEEN

Credit: <https://converseen.fasterland.net>

Batch convert and optimise your images

Nick Peers reveals how to quickly apply the same changes to multiple images at once with the simple-to-configure Converseen.



OUR
EXPERT

Nick Peers doesn't want to think about the fact it's now 40 years since he first sat at a computer. Twenty of those have involved Linux in some form or other.

Ever found yourself in possession of a group of images that you need to resize, convert or rename? If the idea of having to tackle this process one image at a time fills you with dread, then *Converseen* (<https://converseen.fasterland.net>) is for you. It's a batch-processing tool that will revolutionise the way you apply basic edits to multiple images in one go. Simply select your photos, choose what tweaks to apply and let the program do all the hard work for you.

One case scenario is when tasked with providing smaller thumbnails of a group of images for a website. By using *Converseen*, you not only speed up the process but ensure your edits are applied consistently.

Setting up

Converseen can be found in most major distro repositories, but is also available via Snap, Flatpak and as an ApplImage. If you'd like to use *Converseen* to extract pages from PDF files as individual images following our step-by-step guide, we recommend sticking to the older (0.9.9.4) build offered through Ubuntu's repositories: `sudo apt install converseen`. Otherwise, consider the Snap store version (`sudo snap`

install *converseen*) or portable ApplImage for the latest version – 0.9.9.8 at time of writing.

Visit <https://converseen.fasterland.net/download-for-linux/> for all download links and installation instructions on other distros. As with all ApplImages, right-click the downloaded file and choose Properties > Permissions tab to tick All Executing File As Program.

Open *Converseen* from its Launcher shortcut – a two-paned window opens. The main pane contains the images you wish to process – click Open Images to work with single images, or Add Images to add a group. Once the files are added, select one from the list and a preview should appear in the Actions Panel on the left.

Beneath the preview is some information about the image – its type and file size, plus image size (in pixels) and resolution (in pixels per inch, or ppi). Finally, beneath this is a pair of tabs with tools for editing your image: its size and resolution, rotate and flip options, plus output options (including the ability to rename it).

Resize and rescale

Start with the Dimensions tab, which has all the tools you need to resize or resample your images. Tick Scale

CONVERT PDFS TO MULTI-IMAGE FILES

```
nick@ubuntu:~$ sudo apt install imagemagick
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  gfonts imagemagick-6-common imagemagick-6.q16 libaom3 libdav1d5
  libdvdnav3 libdvdread5 libheif1 libltdl7 libmagickcore-6.q16-6-extra
  libmagickwand-6.q16-6-extra libnetpbm10 libopenexr25 libx265-199 netpbm
Suggested packages:
  imagemagick-doc autotrace curl enscript ffmpeg gimp gnuplot grads
  gp2xx html2ps libwmf-bin mplayer povray radiance texlive-base-bin
  ufw-av-batch libfftw3-bin libfftw3-dev inkscape
The following NEW packages will be installed:
  gfonts imagemagick imagemagick-6-common imagemagick-6.q16 libaom3
  libdvdnav3 libdvdread5 libheif1 libltdl7 libmagickcore-6.q16-6-extra
  libmagickwand-6.q16-6-extra libnetpbm10 libopenexr25 libx265-199 netpbm
0 to upgrade, 20 to newly install, 0 to remove and 0 not to upgrade.
Need to get 12.4 MB of archives.
After this operation, 50.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

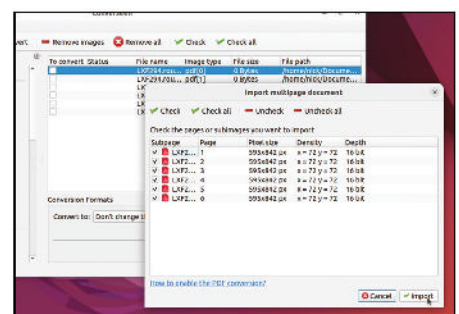
1 How it works

Converseen requires *Ghostscript* (`sudo apt install ghostscript`) and *ImageMagick* to import images from PDFs. If you install *Converseen* via the default Ubuntu repo, you can install *ImageMagick* with `sudo apt install imagemagick`.

```
GNU nano 6.2 /etc/ImageMagick-6/policy.xml
<!-- policy domain="system" name="shred" value="7" -->
<!-- not needed due to the need to use explicitly by mv: -->
<!-- policy domain="delegate" rights="none" pattern="HV" -->
<!-- use curl -->
<policy domain="delegate" rights="none" pattern="URL" />
<policy domain="delegate" rights="none" pattern="HTTPS" />
<policy domain="delegate" rights="none" pattern="HTTP" />
<!-- in order to avoid to get image with password text -->
<policy domain="path" rights="none" pattern="g" />
<!-- disable ghostscript format types -->
<policy domain="coder" rights="none" pattern="PS" />
<policy domain="coder" rights="none" pattern="PS2" />
<policy domain="coder" rights="none" pattern="PS3" />
<policy domain="coder" rights="none" pattern="EPS" />
<policy domain="coder" rights="read|write" pattern="PDF" />
<policy domain="coder" rights="none" pattern="XPS" />
</policy>
```

2 Configure ImageMagick

Type `sudo nano /etc/ImageMagick-6/policy.xml`. Scroll to `<policy domain="coder" rights="none" pattern="PDF" />` and change it to `<policy domain="coder" rights="read|write" pattern="PDF" />`. Press Ctrl+X. Y. Open *Converseen*.



3 Import images

Go to File > Import PDF File. Select a file and click Open, then wait – if you get a non-responding error, keep clicking. Wait until the list is displayed. Tick those you wish to import, then click Import. Choose PNG or JPEG format before processing.

Image to resize the images using the Width and Height boxes. By default, you resize in percentage terms, but when you want to resize all images to a consistent size, click the % drop-down to choose PX, which enables you to set a precise width and/or height in pixels.

Most people resize images proportionally, so tick Maintain Aspect Ratio, which enables you to set the width, then trust Converseen to set the height in proportion to it automatically. If all your images are the same aspect ratio, tick Link Aspect With Selected Image, too – doing so automatically updates the height ratio alongside the width.

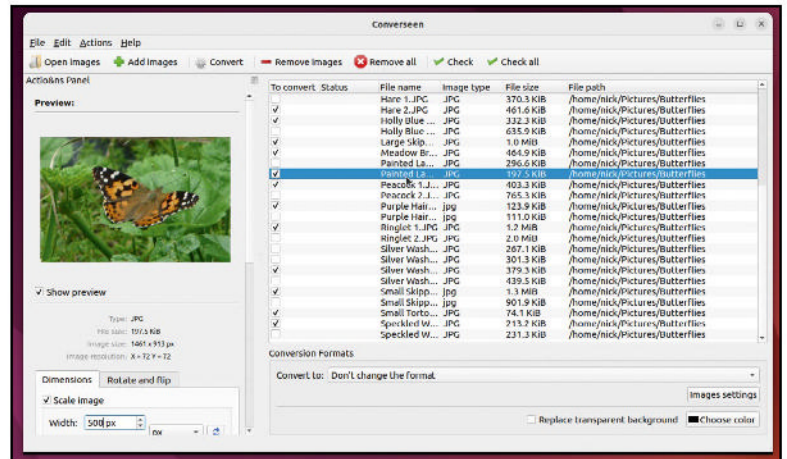
Beneath this is a Resolution box; tick this to change the pixels-per-inch (PPI) ratio of your pictures. This is largely a legacy feature from when people liked to print pictures – the higher the PPI, the smaller the picture (but the better its quality when printed). Ultimately, the quality of your prints comes down to the number of pixels in them, but if changing an image's resolution is important to you, you can set both X and Y figures here (typically they're identical, though) – 300 is usually considered the best option for printing images, 150 for scanned text, and 72-96 for images displayed digitally.

Rename files

Beneath the two tabs lies an Output Options section. Here you can select where to save your converted files – by default, they're saved into the same directory as the originals, so untick Image's Directory and click ... to select a new location.

Below this are options to either rename the files or to tick Overwrite All to replace the originals with your converted files without a failsafe prompt to rename the converted file (not recommended – it's far better to save a copy in a separate folder, review the results and only then manually remove the originals if they're no longer needed).

Tick the Rename box and you have a choice of two renaming conventions: Prefix/Suffix defaults to #_copy, which means each converted file appends _copy to the original filename. Examples of alternatives might include #_thumb (such as filename_thumb.jpg) or converted-# (such as converted-filename.jpg). If



this doesn't float your boat, select Progressive Number and choose a start point (1 being the default).

Format Converseen

If you're looking to convert from one format to another, the Convert To drop-down box beneath the file list is your next stop. Click this and you see a staggering array of supported file types, including image formats and video streams, enabling you to use Converseen to quickly convert from one video format to another.

Beneath this is an Images Settings button – click this if you're outputting JPEG or PNG images. You'll see compression sliders, one for each file type, plus an interpolation filter drop-down (see the box below).

Finally, if your original image has a transparent background, you can replace it with a solid colour by ticking Replace Transparent Background if converting to another format that supports transparency; for formats that don't support transparency (including JPEG), click Choose Color to select which colour to replace the transparent pixels with – the default is black, but you can pick what you like (typically white).

Once you've configured your options, make sure you've ticked To Convert next to each image you'd like to process (a handy Check All button exists if needed), and finally click Convert to process each one. **LXF**

Generating dozens of thumbnail copies of your best images to share or showcase just got a whole lot easier, thanks to Converseen.

QUICK TIP

Converseen can also be used to quickly flip and rotate images. It's a simple case of choosing the appropriate radio boxes under the Rotate And Flip tab: No Rotation, 90 Degrees Right or Left, or 180 Degrees; No Flip, Horizontally or Vertically.

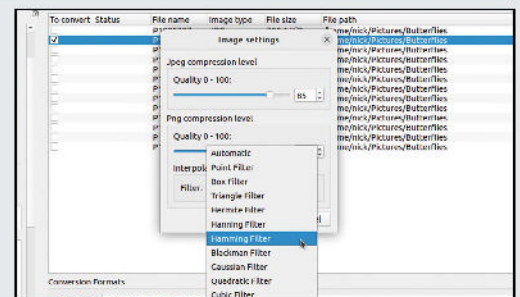
» FIND A FILTER

When you resize images, a process called resampling is used in an attempt to determine the correct values for the pixels in the new image. Shrinking images is simpler because you're not having to guess detail that doesn't already exist, so developers have developed filters to try to correctly recreate how an image would look when blown up.

By default, Converseen applies an automatic filter when resizing images, which is usually fine when making images smaller. However, the program also offers no fewer than 15 individual filters that resample your image in different ways using a variety of

methods. These can be divided into several categories: Interpolated, Gaussian Blurring and Windowed.

No single filter works best across all images, and all come with drawbacks, which manifest in unsightly effects such as ringing (over-compensated edges) or blurring. While you can simply experiment by choosing different filters and seeing what happens, you can get a more informed view by heading over to <https://imagemagick.org/Usage/filter/> – here you'll find a comprehensive guide to all the filters offered by Converseen,



To see which filters are available (and switch between them), click Image Settings followed by the Filter drop-down menu.

grouped into their respective categories. It's a long but worthwhile read if you need guidance on which filters might work best for a particular image.

TEMPLEOS

Credit: <https://templeos.org>

Giving praise to the TempleOS

Michael Reed delves into one of the most unusual operating systems he's ever seen and explores the story behind its creation.



**OUR
EXPERT**

Michael Reed is a Linux veteran and was raised on lesser-known operating systems such as RISC OS, AmigaOS and OS/2.

TempleOS and its modernised derivative ZealOS are unusual operating systems that don't fit into the modern paradigm of how OSes are expected to look or work. Every aspect of the system, from the external user interface to the nitty gritty of how it works on the inside, defies convention.

The man who created TempleOS, Terry A Davis, was affected by serious mental illness problems, and many of his design decisions about the OS were informed by his eccentric views, including his unusual religious beliefs. Throughout his adult life, he had run-ins with the law and had difficulty supporting himself. He often got into spats on online forums and websites. However, he single-handedly created a complete, if unusual, OS worthy of exploration. Sadly, he died in 2018.

There have been attempts to continue the legacy of TempleOS. ZealOS is a modern implementation, and it's another system that we're going to look at.

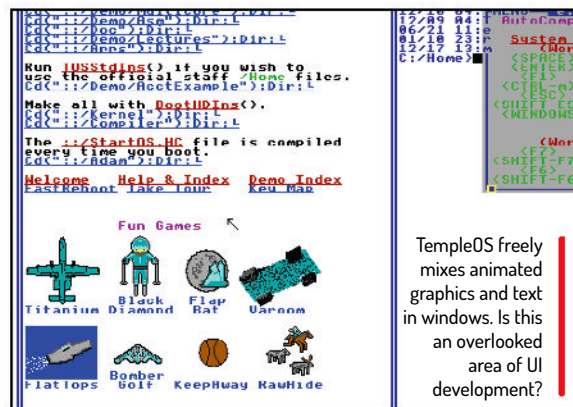
Obtaining and installing

We'll start our investigation into TempleOS (<https://templeos.org>) in the usual way, by obtaining it and installing it. TempleOS is supplied as an ISO that boots to a live environment from which it is possible to install the operating system to a hard disk drive. On launching the ISO, you are given your first taste of the TempleOS aesthetic, which is loud, unconventional and largely text-based. Even at this early stage, things deviate from what you might be used to seeing in an installer, and it takes a minute to orientate yourself and start to navigate the interface.

When you've got over the sensory overload, it's time to choose the installation partition. If you answer in the affirmative when asked if you are installing in a VM, the installation wizard is slightly simplified, but we should warn you, this gives it something of a hair trigger when it comes to the destination partition, so be careful. Other than that, the installation is fairly quick and simple. At the end, you remove the installation media and accept the option to reboot, and the actual boot takes only a few seconds.

First boot

The installation procedure of TempleOS should have given you a rough idea of what to expect from the user



TempleOS freely mixes animated graphics and text in windows. Is this an overlooked area of UI development?

interface. We recommend saying yes to the offer of a tour. This takes the form of an interactive tutorial, and as TempleOS breaks with so many modern GUI conventions, you'd be unlikely to fully fathom the basics of the interface with experimentation alone. As an aside, interactive in-system tutorials have largely died out these days, having been replaced with video tutorials. You can revisit the help window that leads to these tutorials by pressing F1 at any time.

In some respects, the TempleOS interface looks like a text mode display, but it supports simple 16-colour graphics as well as text. It's a multitasking system, with each task running inside its own window. As is the case with a typical desktop GUI operating system, these windows can be moved around with the mouse. The colour scheme is, frankly, a little garish. It reminds us of the text displays of the 8-bit era of home computers and also of some DOS-era PC software. In machines of that vintage, there sometimes wasn't much subtle variation in the colour palette, and it's obvious the creator of TempleOS was heavily influenced by that period of machines and software.

Much of the interface design approach unifies concepts that are normally kept separate in a GUI environment. For example, within a directory listing in a text window, the cursor can be moved around and files can be selected. It's also possible to click on files in such a listing with the mouse, meaning that a directory listing at the command line is also a file and program launcher. This seems unusual at first, but we're left

QUICK TIP

There is no simple way to change the resolution or font size in TempleOS. To keep things as readable as possible when running TempleOS on a virtualiser, consider switching on screen scaling and stretching the display window of the virtualiser, at the cost of some sharpness.

wondering, would it be possible to add some extra interactivity with files on the Linux command line?

Taking things even further, graphics are sometimes thrown into the text environment. Some of these graphical elements can be interacted with, like icons in a GUI, and some are animated.

Scripting and language

As is the case with most OSes, there is a connection between the scripting language and the command line. TempleOS takes this further and integrates the command line, scripting language and programming language. All of these use a C-like language that was created especially for TempleOS, called HolyC.

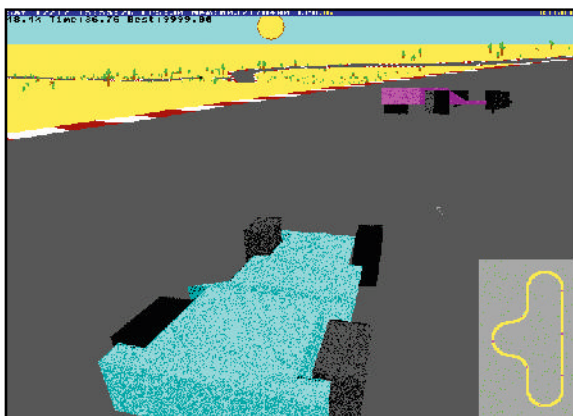
In C-style languages, you can call a function with a statement such as: `printf("Hello");`. For ease of use, the TempleOS command line allows slightly shortened commands, so `Dir;` lists the contents of the current directory. As it uses C-style syntax, you can combine commands on one line. For example, `Cd("/home");Dir;` changes to the **home** directory and lists its contents.

You might think this means that HolyC is a scripting language, but it's actually a compiled language like C. Behind the scenes, the TempleOS HolyC compiler interprets and compiles each line that is entered at the command line. One advantage of this approach is that it simplifies the process of learning the command line, scripting language and programming language.

Much of the system is implemented in this way. In most systems, utilities and programs are stored as object code (the result of compiling source code) or a scripting language, but TempleOS blurs the distinction as most programs and utilities are stored as source code and compiled and run as needed. One advantage is that the program source code can be edited at any time. It does mean program startup might be delayed, but it isn't noticeable for smaller programs. Unlike implementing programs using a scripting language, programs execute at full speed when running.

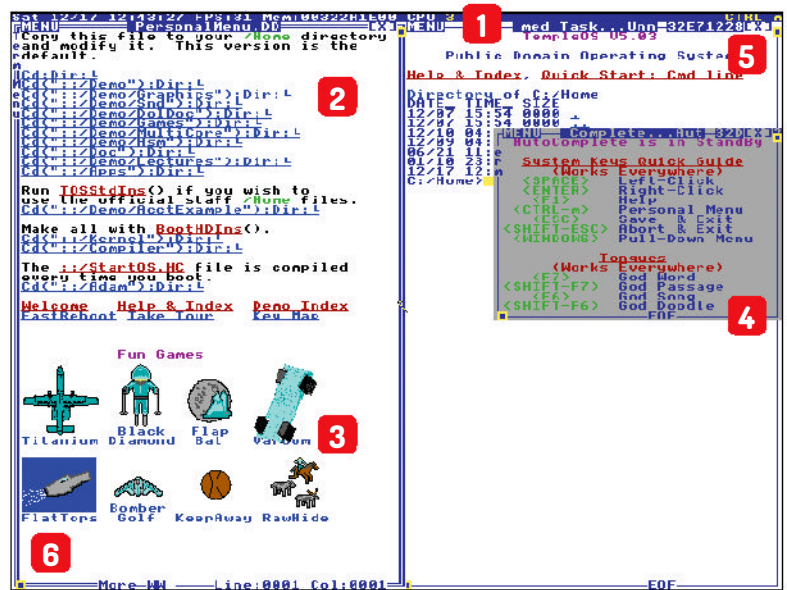
Under the hood

In terms of architecture, what approaches does TempleOS use? Like Linux, it uses a monolithic kernel design, meaning that components such as drivers exist in the same space as the kernel. It's a 64-bit kernel, which has a small performance advantage over older, 32-bit kernels, but more importantly it can access huge amounts of memory and process enormous files and



Driving game Varoom is one of the more playable example games that come with TempleOS.

EXPLORE TEMPLEOS



1 Status area

This part of the desktop lists things such as the current frame rate of the screen and CPU core utilisation.

2 File use

When files are listed in a text window, they can be clicked on with the mouse or selected with the cursor keys.

3 Fun and games

These icons are animated and can be clicked on. Categories include Fun Games, Unfun Games and Not Games.

4 Shortcut guide

This pop-up window serves as a quick guide to always-available system function keyboard shortcuts.

5 Windows

TempleOS supports multiple windows that are moved in the familiar way, by dragging the title bar.

6 See more

If the content of a window is larger than the window, there is a More prompt at the bottom. Scroll down using the cursor keys.

filesystems without having to resort to breaking those resources up into smaller parts to work with them.

As well as being a 64-bit OS, TempleOS has support for multiple CPU cores. From hereon in, the design is a bit unusual for a fairly modern operating system. However, it's worth bearing in mind that TempleOS has an experimental air about it, and it was never designed to be a general-purpose operating system.

Multitask multithread

Rather than offering preemptive multitasking like systems such as AmigaOS and Linux, TempleOS uses cooperative multitasking – the applications themselves have to hand control back to the operating system periodically. This leads to multitasking that isn't as smooth, and it presents problems when trying to execute background tasks. It can also result in a system that isn't as reliable, because a crashing app can bring the entire system down. We highlighted this as a shortcoming of RISC OS in a previous article.

Tasks can launch their own subtasks, meaning TempleOS does, technically, support multithreading. However, it doesn't enjoy the full benefits of that as it doesn't support preemptive multitasking, so threads can't execute independently and simultaneously.

Memory protection

We expect a modern OS to offer memory protection so user mode programs can't interfere with the memory space of other programs or the kernel. TempleOS

QUICK TIP

TempleOS programs are launched by either clicking on them or including them in the current environment: `#include <path and name of program>` loads and executes a program. It's another part of TempleOS that doesn't work like other systems.



The Temple Operating System

Downloads

Terry's Linux Bash Scripts
[General](#)
[QEMU](#)
[VirtualBox](#)
[VMware](#)

Screen Shots

Introduction

Features

Source Code By File

Instructional Videos

F.A.Q.

The Charter

Source Code (Raw)

Some of the TempleOS website is looking worse for wear in terms of layout, but it's still navigable, and it's packed with information and Davis's personal insights.

QUICK TIP

Pressing F7 at any time invokes the Tongues Function of TempleOS and copies a random word to the screen. Pressing Shift F7 inserts a random Bible quotation. Pressing Escape usually takes you back to the previous screen.

doesn't work like this. Part of its design ethos is that these restrictions make the system more difficult to program for and to understand. This is in line with the architecture of home computers of the '80s, which were a big influence on the TempleOS design.

On those old home computer systems, the programmer could poke and peek anywhere they wanted, which encouraged experimentation. Indeed, creator Terry A Davis once described TempleOS as a Commodore 64 with multicore x86 support.

The snag is that when you have a system that can run more complex programs, or multiple programs at once, a lack of memory protection could cause havoc. These problems are multiplied when you take a computer online. A lack of memory protection has always been a shortcoming of AmigaOS, to give one example. However, as TempleOS deliberately offers no networking facilities, this latter point is moot.

Graphics system

TempleOS has a graphics library that can produce 2D and 3D graphics. There is no hardware graphics

acceleration, but Davis felt this might be compensated for by the support for multiple CPU cores. As it stands, TempleOS is limited to 640x480 resolution with 16 colours. Unusually, it updates the entire screen 30 times per second and merges the graphics and text layers, creating another barrier to higher resolutions.

Developing software

Nearly all the software available for TempleOS was created by Davis himself. It isn't a particularly viable target system for ports. The principle programming language of TempleOS, HolyC, is similar to C, but Davis made various customisations. TempleOS doesn't feature any compilers for mainstream languages or frameworks. It is possible to create programs from scratch for TempleOS within TempleOS.

The easiest way to start exploring the example apps included with TempleOS is to press Ctrl+M in a terminal to bring up the Personal Menu. This app launcher can be scrolled up and down using the cursor keys and apps are launched with a mouse click or by pressing Space. As with all of TempleOS, the files and directory listed in this window as text can also be interacted with.

Creation of TempleOS

TempleOS was largely created by one man, the late Terry A Davis. There's no doubt he was an extremely capable programmer. Unfortunately, he suffered from mental health issues for all of his adult life, and was eventually diagnosed with schizophrenia. He had a degree in mechanical engineering and worked professionally as a VAX programmer, but the mental health issues that dogged him often made it difficult for him to remain employed. At times he was homeless and he had some run-ins with the law.

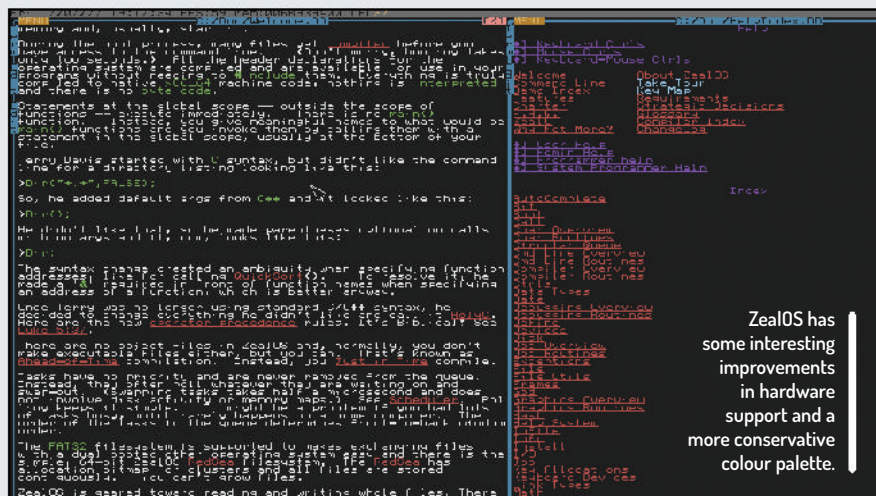
He was a well-known character in the hobbyist OS scene but often fell out with people, and sometimes ended up being banned from online forums. He posted to programming discussion areas of sites such as

» ZEALOS

ZealOS (<https://github.com/Zeal-Operating-System/ZealOS>) describes itself as "a modernised, professional fork of the 64-bit Temple Operating System" and is a continuation of a TempleOS fork.

Offering notable enhancements over the final version of TempleOS, the first thing that hit us about ZealOS was the more conservative colour scheme. It is also possible to select between multiple resolutions with 32-bit colour depth.

One of the biggest changes is the addition of networking hardware support and a full network stack. This particular addition opens up the possibilities of what the operating system can be used for, but it has to be remembered that the design is inherently unstable and insecure. The hardware support has been expanded with AHCI support for storage devices.



ZealOS has some interesting improvements in hardware support and a more conservative colour palette.

TempleOS is fascinating, but we regard ZealOS as the better starting point if you want to pursue TempleOS's

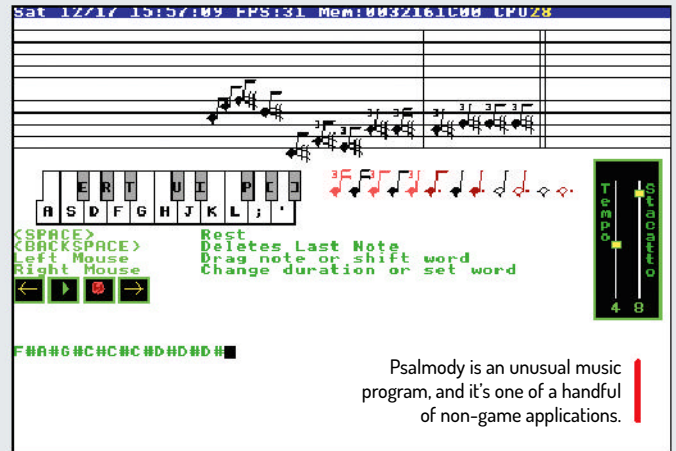
original stated purpose as a "modern Commodore 64" for experimenters and recreational programmers.

» INCLUDED SOFTWARE

TempleOS has a reasonably complete set of system tools, including those needed to navigate around the filesystem. It has its own filesystem, called RedSea, and it supports FAT32 partitions for file exchange with other operating systems. It also has a plain text editor and a programming editor with syntax highlighting.

The several games that are included with TempleOS are, humorously, divided into the categories of Fun Games, Unfun Games and Not Games. We regard these games as programming examples, and they remind us of the sort of programs that people would type into home computers from listings in magazines in the '80s. That said, they do take some advantage of the power of a modern CPU, as some of them feature smooth scrolling, rotating play areas and 3D graphics.

Within the directories that are referenced on the Personal Menu, there are dozens of smaller graphical examples. It's clear that Davis wanted TempleOS to offer the free and joyful experimental programming fun of 8-bit home computers, such as his Commodore 64 that he originally



learned to program on. It's certainly true that, by comparison, modern programming environments tend to get in the way of casual experimentation.

Reddit, Something Awful, OSDev, HackerNews, Stackoverflow and OSNews, and he posted on social media sites such as Twitter.

Davis's religious beliefs were a major influence. Although there were practical considerations behind the choice of such a low resolution and colour depth, for example, Davis also stated that the decision was due to commandments sent to him from God. He would regularly claim to be in contact with God, and his postings on this matter ranged from aggressive to satirical to incomprehensible. There are some Bible-based functions within TempleOS, and many of the included games are religiously influenced.

His streaming videos and online posts attracted attention from various communities. Many of these people were sympathetic, but some individuals took to cruelly provoking him. There are some saddening videos online of him being provoked by people who'd ring him up on the phone while he was live streaming.

Over the course of 2017, Davis's condition worsened. He video-blogged frequently. In the videos of that era, he is clearly confused, and his speech is strewn with references to bizarre conspiracy theories and frequent racial slurs. He spent most of his last two years living in a van or homeless on the streets. Some of his supporters would donate money to him.

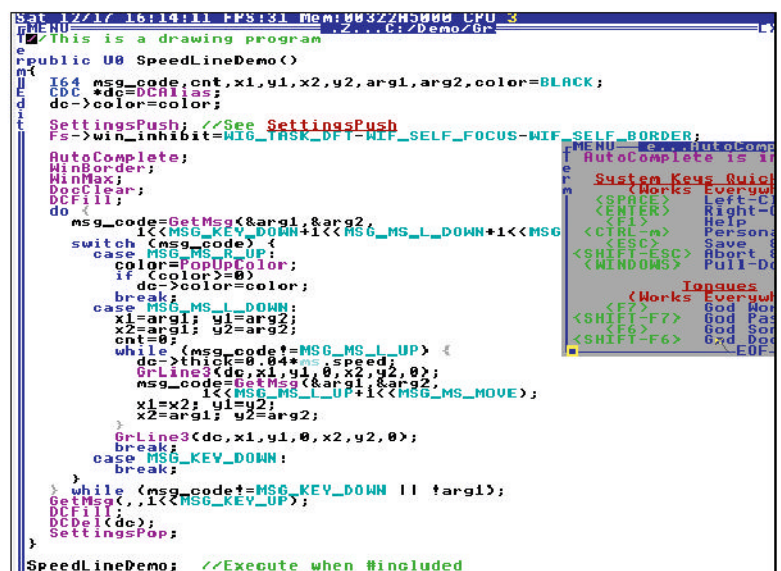
While travelling around the US in 2018, Davis was struck by a train and killed. It is uncertain whether this was a deliberate act of suicide or an accident. Subsequently, various tributes were made by different online communities, mostly in admiration of his technical skills and humour, while sympathising with his mental health problems. TempleOS remains as a final tribute to those skills and his unique way of thinking.

TempleOS worship

It's clear that TempleOS was never intended to be a primary, general-purpose operating system, and this

point was often emphasised by Davis himself. In some ways, this is a more sensible goal than some hobbyist operating systems because, realistically, those systems are unlikely to ever take over from the user's main operating system.

We would recommend exploring TempleOS as a fascinating experience rather than with any particular goal in mind. A continuation, such as ZealOS, is probably a better bet if you intend to use TempleOS for its intended purpose of recreational programming. We have heard of the concept of programming as art, but this is an operating system shaped by the eccentric world view of Terry A Davis. Exploring his operating system is – like trying to understand the man himself – often confusing but constantly both interesting and thought-provoking. **LXF**



Practically all of the example programs and system elements are sitting around as source code, ready to be explored.

» IMPROVE YOUR LINUX SKILLS Subscribe now at <http://bit.ly/LinuxFormat>

BACK ISSUES » MISSED ONE?

ISSUE 298 February 2023

Product code:
LXFDB0298

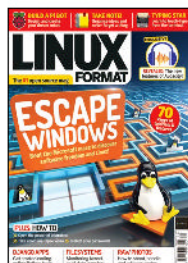


In the magazine

Dig in to our New Year smorgasbord of hot Pi projects, from Pico devices running MicroPython to full-Pi setups. Plus, discover how you can contribute to open source projects without being a coder. We also have a *Roundup* of Matrix messaging clients, and a torrent of tutorials on everything from tackling tables to planning a novel, polishing video footage, using *GIMP* and lots more besides.

ISSUE 297 January 2023

Product code:
LXFDB0297



In the magazine

Escape from Windows by setting up and using the extremely user-friendly Elementary OS. We also help you take pro-level shots with our guide to RAW photography and present the inside scoop on the future of *Audacity*. There's a *Roundup* of the best note-taking outliner apps, too, plus tutorials on touch-typing, password management, building a Django app and a whole lot more.

ISSUE 296 December 2022

Product code:
LXFDB0296



In the magazine

Stay safe and protect your privacy online with our *Firefox* guide. Plus, discover what the future holds for 3D printing and how to upgrade to the latest version of Ubuntu 22.10 – Kinetic Kudu. Elsewhere in the mag we look at five vector drawing apps, review Raspberry Pi OS and have tutorials on recreating retro platform games, building static websites, creating a cataloguing app and much more.

ISSUE 295 November 2022

Product code:
LXFDB0295



In the magazine

Relive past computers and games with our in-depth look at retro emulation. We also round up five lightweight distros, step into the exciting world of Windows Subsystem for Linux, and chat to Alex Jones, director of Kubernetes Engineering at Canonical, about Kubernetes and the cloud-native future. There are tutorials on internet bandwidth, PC search and loads more, too.

ISSUE 294 October 2022

Product code:
LXFDB0294

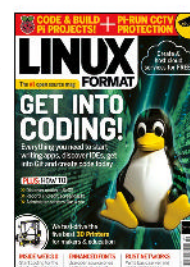


In the magazine

Get your system running the latest Mint, with other distros and even Windows! Elsewhere, we assess five creative writing tools, show how to keep the lights on in your smart home, build a Raspberry Pi media centre, turn smartphones into live cameras, perfect 3D prints, create better cut-outs in *GIMP* and recreate Pac-Man in Python. Plus we reveal the latest AlmaLinux developments.

ISSUE 293 September 2022

Product code:
LXFDB0293



In the magazine

We explain coding principles, what editor to use, how to use Git, and help you take your first steps writing your own code and controlling physical Pi projects. We review five of the best 3D printers, put together a CCTV system, assess a recreation of the BeOS operating system, and show how to create custom fonts. There's also tutorials on Web 3.0, screencasting and cloud hosting.

To order, visit **www.magazinesdirect.com**

Select **Single Issues** from the tab menu, then select **Linux Format**.

Or call the back issues hotline on **0330 333 1113**
or **+44 (0)330 333 1113** for overseas orders.

Quote the product code shown above and have your credit or debit card details ready.

USA? EU? THE MOON?

UK readers
turn to
p16

SUBSCRIBE!

Don't wait for the latest issue to reach your local store – subscribe today and let *Linux Format* fly straight to you. Faster, cheaper and with DRM-free archive access!



**3 GREAT
WAYS TO
SUBSCRIBE**

Print, digital-only,
and print+digital
bundles!

» **USA**
From \$132
For 13 issues

» **REST OF THE WORLD**
From \$132
For 13 issues

» **EUROPE**
From €100
For 13 issues

IT'S EASY TO SUBSCRIBE!

Visit www.magazinesdirect.com/linux-format

Call **+44 0330 333 1113**

Lines open Monday-Friday, 9am-5pm, UK time.

*We don't actually deliver to the Moon. Yet.

Using enhanced layer features with G'MIC

Karsten Günther takes a look beyond GIMP's standard layer functions. Besides special GIMP operations, tasks provided by G'MIC loom into view.



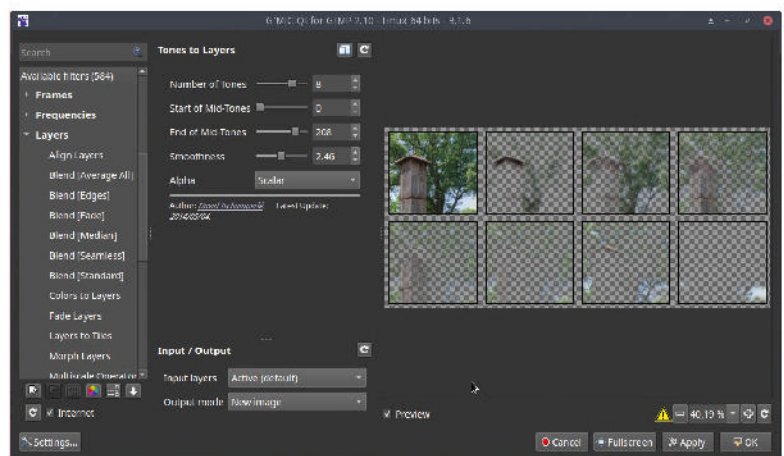
**OUR
EXPERT**

Karsten Günther has written entire books about using GIMP, so four pages is just a morning's work to him.

Both GIMP and the plugin G'MIC have a number of layer functions that – in contrast to those in the article from **LXF298** – are neither simple nor obvious. We'll examine them in more detail here. With G'MIC, it usually looks like this: a number of functions available with GIMP can be found with extended possibilities and usually there are a range of additional functions that offer something similar. Some of these functions are rather theoretical, but many are quite practical. The layer-specific functions are mainly found in the Layers, Sequences or Colour sections, and scattered in the Testing section.

Separate colours

GIMP supports the idea of separating colours in layers with the Decompose functions described in the tutorial in **LXF298**. G'MIC extends these possibilities with the



Tone values in difficult exposures can be specifically influenced with Tones to Layers. Due to the division into layers, far-reaching possibilities – including layer modes – are available.

Channels To Layer filter, which enables you to separate channels in layers instead of RGB, CMY and (limited) HSV. However, these functions work somewhat differently from GIMP's own. For example, the RGB layers have to be combined with the Lighten or Screen mode to get a colour-correct image again. For CMY

» G'MIC ON THE COMMAND LINE

G'MIC is currently being developed in two variants in parallel: as a plugin for GIMP (`gimp-plugin-gmic`) and as a command-line program (`gmic` and `gmic.qt`). Both variants differ only in details, adapted to the way they are used. For example, the command-line program has a demo mode that would make no sense with the plugin; `gmic demos` starts this special mode.

Without specifying an option or function, `gmic` works as a universal display program; `gmic tst.png` displays the image. Normally, however, the command is called with both an input

file and a function. The option `-h` outputs extensive help. To write the output to a file, the option `output TST.jpg` is available.

On the command line, G'MIC can convert the RGB data of images into very different colour models. For example, `gmic IMAGE.jpg rgb2cmk split c` calculates the CMYK conversion and displays the result. If `-output /tmp/tst.png` is added to the line, G'MIC writes LAB, LCH, HSL, HSV and a number of other, less well-known colour models. Note that GIMP and G'MIC use different formulas for the conversions in detail.



Interactive and special G'MIC functions are provided by G'MIC demos. As the name suggests, they are mainly used to demonstrate G'MIC capabilities.

layers, the Difference mode is preset. With HSV splitting, G'MIC creates only two layers: one greyscale layer with the lightnings and one with the colours.

As a result of this colour splitting, there are ways to finely influence the individual layers with the gradation curves (GIMP's Curves tool). Both colour casts and brightness can be edited after. To merge layers created with this G'MIC filter back into one after editing, you must edit the layer stack from below – first the lowest two, then the layer above – with Merge Down.

Colors To Layers is a special G'MIC variant of the filters for separating colours. In this case, G'MIC directly uses the RGB values without conversion to distinguish the colours. All three parameters have a direct influence on the result. If the preview in this filter shows only two layers, the value for Minimal Area (%) is often set too high. If the maximum permissible number of layers is not reached, this can be changed via Colour Tolerance. Whether Autocrop Output Layers is useful depends on the use of the filter. By default, it creates layers in the image size. As this allows up to 256 output layers, quite fine colours can be distinguished.

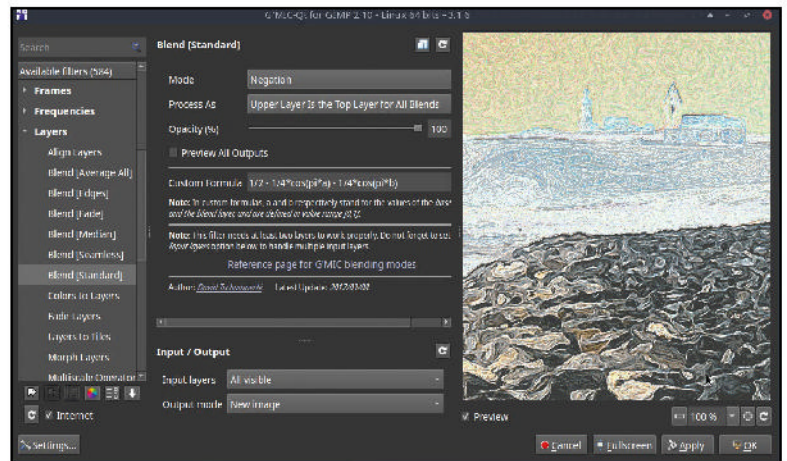
Tones To Layers is another variant. At first glance, this filter seems bulky and not very useful. However, its usefulness is apparent in practice. The separation takes place in a residual layer and up to 10 tone layers. The Mid-Tones control adjusts how the split is done; the transitions are controlled by Smoothness. Higher Smoothness values increase the areas that end up on a tone level. Together with the residual layer, the tone layers make up the entire image. By default, the layers are combined with the Normal mode.

For individual tone layers, different layer modes can be set, so the overall effect can be specifically adjusted. The tone layers can also be individually doubled or their opacity can be reduced to manipulate details in shadows and highlights, for example.

Blending

Many of G'MIC's layer filters provide quite different blending functions. The effects of the blending filters range from extended layer modes to complex copy processes. With these, several input layers must be selected to achieve an effect. Usually, the All or All Visible options for Input Layers are a good choice.

Blend [Average All] is basically a filter based on the GIMP functions that averages the pixels of the



selected input layers and outputs the result. In the case of layers with completely different contents – that is, ones that were not created as copies of one layer – the result looks like the multiple exposures of a film. The three colour models provided in the filter display quite different results, which shows their necessity. The blending is done in a soft way that blurs edges and contours. With more or less identical layers, the filter hardly shows any effect.

A variant of this filter with a different, but similar algorithm for calculating the pixels is Blend [Median]. Here, too, the parameter Colourspace controls the colour model used and thus influences the effect.

The filter Blend [Edges] has a similar function, but has different parameters. This filter often shows relatively pronounced edges and contours in the multiple exposures and is thus clearly different from the filters described above.

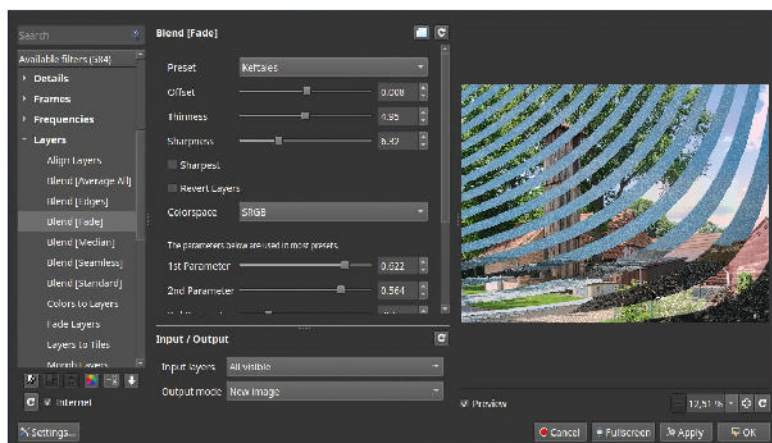
GIMP's Layer Modes simulates Blend [Standard], but adds a whole new set of interesting modes, such as Negation and XOR. How the calculation for the preset modes is done is shown on the given web page (see *Quick Tip, below-right*). Most of the parameters of the filter are immediately understandable, but there is no selection of the colour space used. Therefore, all operations are done in the SRGB colour space. Process As defines the way G'MIC processes several layers. With two layers, the order of processing is trivial, but becomes more demanding with three layers. Here there are several possibilities, each of which produces different results. Preview All

Outputs helps to understand the effect of the Process As settings.

A really special feature of this filter is the Custom Formula mode. It offers the possibility to define your own formula for the calculation. In this formula, the variable **a** stands for the pixels of the base layer (lower layer) and **b** for the layer above (blend layer). The values are normalised to the range [0..1].

The Blend [Fade] and Blend [Seamless] filters have a completely different effect from the Blend filters discussed previously. Blend [Fade] places templates that act

The G'MIC filter Blend [Standard] provides extended layer modes. They extend the modes available in GIMP by, among other things, the possibility to use own formulas for the calculation.

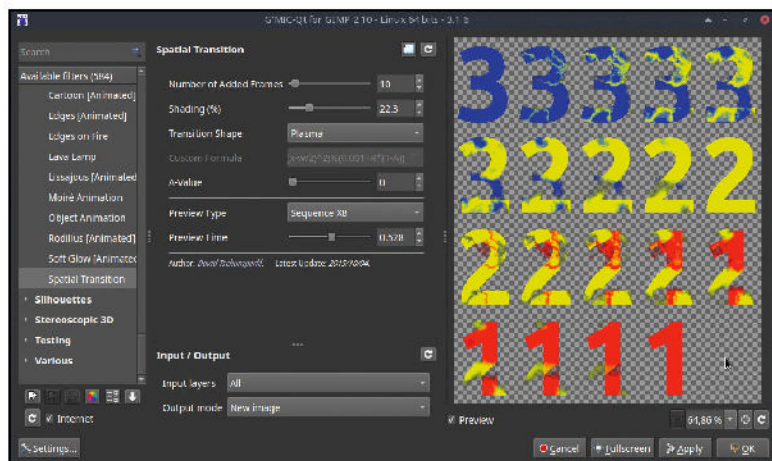


Blend [Fade] creates overlays that act like templates. A custom formula, which can be defined by the user, takes up the mask and enables a variety of patterns.

QUICK TIP

The blending or levels modes are relatively simple formulas. The ones used by G'MIC can be found here: <https://bit.ly/lxf299formula>, while GIMP summarises them here: <https://bit.ly/lxf299layers>.





Transition effects such as those used in presentations are created by Spatial Transition. The filter is customisable within a wide range by means of a custom formula.

QUICK TIP

If the animations are to be able to run in an endless loop, the first and last entrance layer should be identical to prevent a hard transition.

as masks over layers and thus creates partially transparent layers. In the holes, the underlying layers shine through. This makes some interesting patterns possible. The three parameters in the lower part of the dialog window control the display. A custom formula is also available for this filter. Within it, the variables x (first parameter) to z (third parameter) can be used.

In addition to this filter, Spatial Transition from the Sequences section should be mentioned here. This filter also creates a transition between two layers and takes into account the alpha channels of the input layers. The type and manner of the transition are determined by the parameters in the upper part of the dialog window. Transition Shape defines the type of transition, Shading and A-Value control the details. Even the default transition Plasma usually creates very interesting effects – it uses GIMP's plasma filter and

random patterns – but the other variants are at least as interesting.

Two special features of this filter are worth mentioning. A custom formula can be defined as a transition shape. Its definition should initially be based on the default setting, for example: $\sin(x*y*0.1\pi/w*(1+100*A))$ or $\sin(y*x/w/h\pi/(1+100*A))$. In the lower part of the dialog window, the preview can be set more precisely. The Preview Type Transition Map, for example, shows the mask used for the transitions. Timed Image shows the state of the animation depending on the Preview Time parameter.

You can read about the Blend [Seamless] filter at <http://gimpchat.com/viewtopic.php?f=28&t=10204>. G'MIC seamlessly blends the upper layer into the layer below. The Inner Fading and Outer Fading parameters control how the fading should be done at the edges. Mixed Mode changes the visibility of the inserted layer considerably, so should be used with caution. With the Output As Separate Layers option, the output is done as an additional layer, which makes post-processing easy. Only in this form is it possible to change the opacity of the inserted layer or manually adjust parts of the layer. G'MIC also shows exactly where and which changes the filter makes to the inserted layer.

Layers inserted into an image with Ctrl+v do not have to be converted into normal image layers for this filter. It is sufficient to activate Active and Below or All Visible as Input Layers. For the output, New Image is often a good choice; it allows unsuccessful experiments to be simply discarded without affecting the output data – that is, the current image.

The URLs given in the dialog window are still current today and show quite excellently what the filter is capable of. However, the application needs much more practice than the examples shown there suggest. It is often not clear why inserted layers fit very well or why they do not fit well. The filter is therefore one of the important tools that one should know (and be able to use), but it requires some experience.

» GIMP AND ANIMATIONS

Animation tools are limited but GIMP is able to directly play animations created with the program or with G'MIC. Also, GIMP can only save animations in two formats: the now completely outdated GIF and the little-known but supported MNG. This format essentially consists of a series of PNG images that can be extracted losslessly. Ironically, GIMP is not able to read this format.

However, MNG files can be converted to other formats on the command line with the ImageMagick command `convert convert tst.MNG tst.mp4` converts the animation into the popular video format. Conversely, `convert -verbose tst.mng png:/tmp/x` splits an MNG animation into frames. Specifying `png:` does this as PNGs.

If you want to have a greater influence on the result, you can take a detour via the animation tool Imagination (<http://imagination.sourceforge.net>). This enables you to generate MP4 animations from individual images that are manually controlled. This really has a far-reaching influence on the representations. The same is, of course, also possible with Kdenlive and probably other video-editing programs, but there the effort is considerably greater.

One more warning about the animations: with multiple input layers and many intermediate layers, GIMP really takes up a lot of RAM space to edit them, with 16GB RAM and swap being quickly used up. It helps to reduce the colour depth set by GIMP when reading images, in the Image menu under Precision.

Align and inform

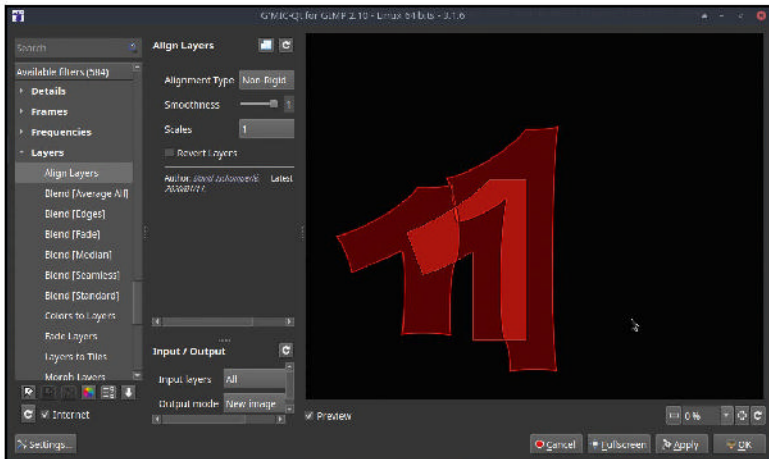
One group of G'MIC filters is not used directly but only indirectly for image composition. These are filters that serve for preliminaries or are useful in organising many layers in an image.

Align Layers is a filter that only works correctly with image layers; with text layers the results are often incorrect. It tries to shift the position of the layers using salient points, so the contents overlap as closely as possible. The Rigid Alignment-Type tries to keep the overlap as high as possible, taking into account the Smoothness parameter. Scales allows multiple sizes of objects when aligning.

Note: with this filter, the preview does not show the result directly but an overlay of all aligned layers, as it is created by Blend [Median] or Blend [Average].

This filter is reminiscent of the shell command `align-image-stack` from the Hugin package, which is used to align several individual images when constructing panoramas.

The Layers To Tiles filter is also extremely simple, but useful – it collects all of the entrance layers and represents them sorted. Two other filters have similar properties: Pack takes all the input layers and tries to



Align Layers tries to align layers automatically. Sometimes it takes a few tries to find the optimal parameters for this. The preview shows an overlay of the layers.

arrange them so that they are packed as thickly as possible. The order can be controlled within certain limits with the Ordered By parameter. This makes the filter useful to obtain an overview of the layers used in an image, or to arrange space-saving objects on layers.

Quite a lot of information about levels can be found within Layer Info (in the Testing section). In addition to the level name, the opacity and position – both in the picture and in the layer stack – there is a lot more information available. This filter is so easy to use and relatively easy to expand that it is well suited as a basis for your own experiments.

Animations

GIMP creates animations as images with a large number of layers. Each single image of the animation is available as a separate layer – and can be reworked individually. GIMP or G'MIC can create animations with special filters that automatically calculate these individual images. GIMP has a few simple filters in the Filter menu under Animation. G'MIC significantly expands the list of these filters. The Layers section contains two relatively simple animation filters. In the Sequences section, there are again almost 20 animation filters that are more complex.

First two simple filters: Fade Layers blends layers into each other. To do this, the filter gradually reduces or increases the opacity of the individual layers. It is applicable for any number of input layers, which then appear one after the other. A special feature of the filter is the fact that it works with layers that have (and receive) an alpha channel.

The Morph Layers filter goes one step further than Fade Layers. With this filter, G'MIC tries to automatically adjust the forms contained in the layers when creating the fluid layers. An example of three text layers (1, 2, 3) shows how far this goes when using this filter. With pictures with more natural content, the results are often even better. However, the filter removes alpha channels automatically from layers, so you should check which background colour is preset before you decide to use the filter. (However, if you are looking for really good control options for morphing,

the Filter Morph [Interactive] in the Deformations section is better for this task.)

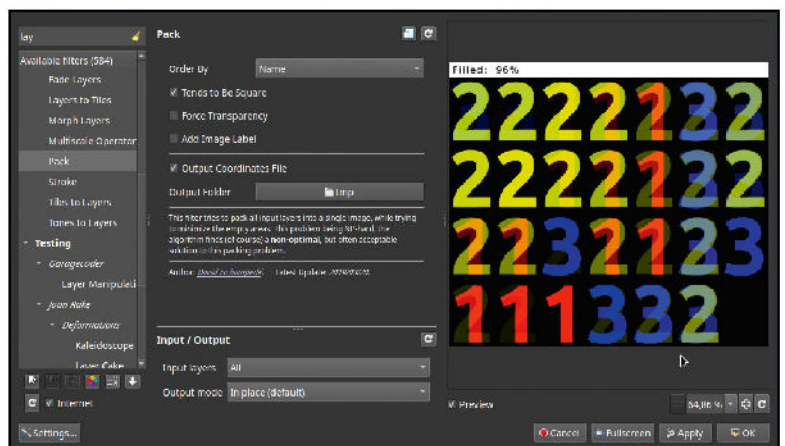
Some of the animation filters in the Sequences pane incrementally create specific representations that filters from other areas create directly. The basis for these filters is always a known filter, whose parameters G'MIC automatically changes step by step. The dialog box and the preview show the start and end points of the editing.

B&W Pencil [Animated] is one of G'MIC's easy-to-use animation filters. The function is based on the B&W Pencil painting filter, which generates a black and white drawing from a colour image. It

creates an animation in which the image develops from fine to coarse. Sometimes, the opposite makes more sense. In this case, the layer stack can be reversed before the animation is saved or played.

Soft Glow [Animated] and Rodilius [Animated] are both based on the non-animated filters of the same names and have only simplified settings. Nonetheless, these filters are extremely easy to use – however, the achieved effects are not particularly impressive. Unfortunately, this also applies to the Edges On Fire filter. This first uses a simple edge filter to find the relevant areas in an image, where animated flames are then created. This can work well, but it is no longer updated.

Worth examining is the 3D Extrusion [Animated]. It is based on the 3D Extrusion filter from the Rendering section. This filter calculates three-dimensional structures from or in images and displays them as a map. It is complex and often – with natural images – not easy to apply. For such images, the relevant parameters should first be determined in the non-animated version. However, it is easy with text layers and clearly defined structures. The animated filter creates the view of a rotating object from the input planes. 



Pack grabs the input levels as compactly as possible and at the same time gives an overview of the existing and selected layers.

QUICK TIP

Unfortunately, the current G'MIC version no longer has the ability to automatically name layers with the functions and parameters used. In order not to lose this information, this should always be done manually first after creating new layers.

» **GET CREATIVE EVERY MONTH!** Subscribe now at <http://bit.ly/LinuxFormat>

ALGORITHMIC ART

Getting creative with algorithmic art

Mike Bedford demonstrates how you can exercise your left and right brain simultaneously by generating algorithmic art.



OUR EXPERT

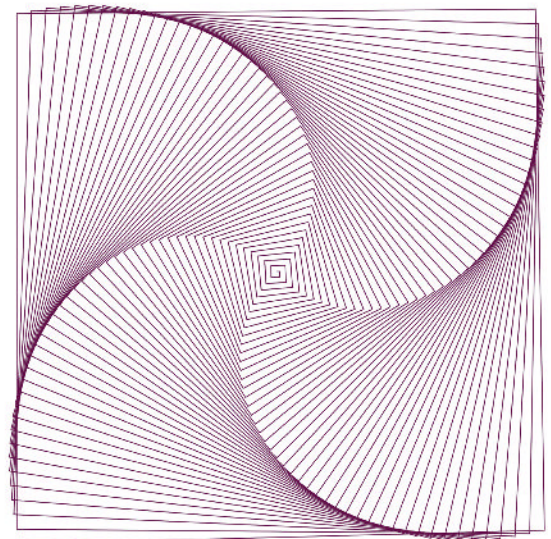
Mike Bedford is firmly of a technical persuasion, yet has loved to exercise his more creative skills since he first got to grips with computer programming.

It has been suggested that creating artwork and computer programming are diametrically opposite. After all, art involves exercising creativity, while coding involves providing a fixed set of instructions that will be followed verbatim. More generally, the divide between the arts and the sciences is often considered unbridgeable. We'd beg to differ as, no doubt, Leonardo da Vinci would have done. As one of history's best-known polyglots, his achievements included the painting of *The Last Supper*, a mural that can be seen in the monastery of Santa Maria delle Grazie in Milan, his writings on a wide range of scientific disciplines including anatomy, botany, astronomy, mathematics and geology, plus his many inventions, perhaps most notably flying machines. With such an esteemed artist-cum-scientist as our inspiration, we feel more than justified in introducing you to the subject of creating art using a sequence of instructions. In some cases, we're looking at how to do that using software that others have created, but if you are itching to write code yourself, we look at how you can do so to create your own unique algorithmic art.

Art has been based on the Mandelbrot set, like this one generated using Fractive, just by zooming in and using imaginative colour schemes

The early days

It seems that, in most cases, the early days of computer-generated art involved very little in the way of creativity. By the early '70s, pen plotters and primitive vector graphics terminals were starting to become mainstream, and programmers were keen to

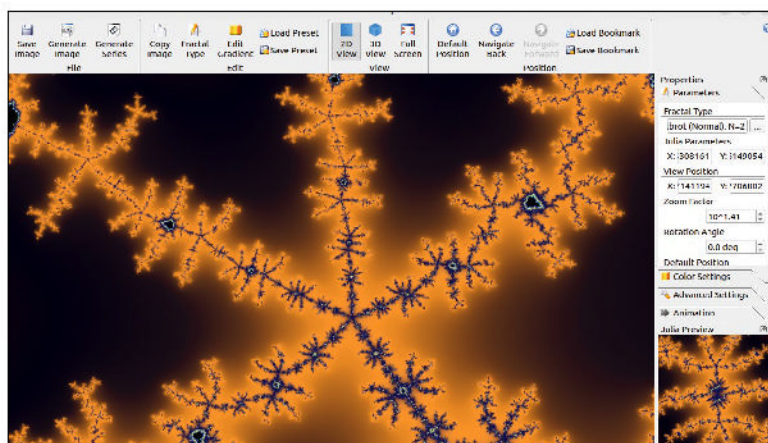


Despite its simplicity, variants of this polyspiral, typical of the early days of purely geometrical computer art, have created attractive art.

investigate the opportunities on offer. Of course, serious uses included plotting experimental results as graphs and creating engineering drawings, but as a first step to learning about this technology, students often experimented with basic geometric manipulations. And, as soon became clear, simple programs could produce some eye-catching designs.

We imagine, many coders are creating much the same designs today. Indeed, we'll look at one way of doing that – and going much further – when we delve into the Processing programming language. As an example of this early form of computer-generated art, though, we present an example of a polyspiral (above), which can be generated with very few lines of code. Is it art? Barely, although we have come across a variant on the theme comprising a matrix of similar polyspirals in a range of different colours, which might not look out of place framed and hanging on a wall. As a case in point, we should mention the geometric Islamic art – symmetry, tessellation and all – that adorns many of the world's great mosques.

If we move the clock forward by a decade or so, from the geometric designs produced using pen



plotters on mainframes and mini-computers to the early days of affordable home computers, fractals including the Mandelbrot set became hugely popular. Since their appeal has now waned somewhat, if you've not come across it before, let's just say that the Mandelbrot set is generated by repeatedly applying a very simple formula, and the result, when displayed graphically, is astonishing.

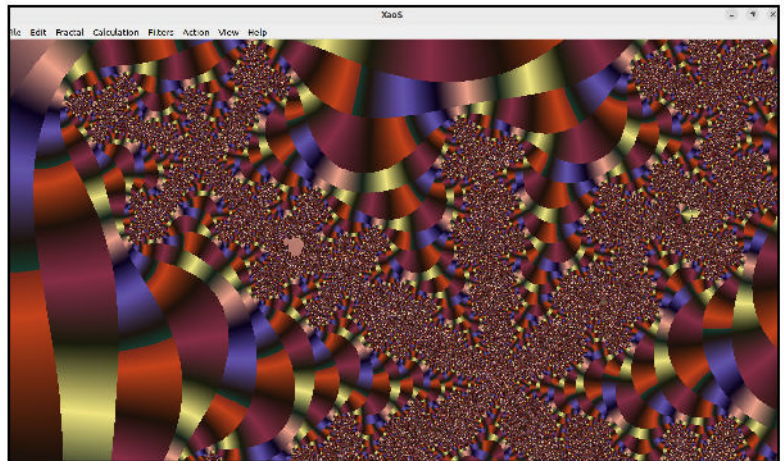
The scope for generating '60s-style psychedelic art isn't difficult to appreciate and, while there's no creativity involved in the generation of these patterns, we might say that the artistic element comes from discovering aesthetically pleasing images from within its infinite depth. In fact, we could draw a parallel with the somewhat tongue-in-cheek suggestion that a sculptor does nothing more than release, from a block of stone, a work of art that was always present. Fractal artists are still with us today and creating some remarkable artwork, but now the creativity often comes from devising ways of manipulating the basic fractal image in a myriad of ways.

Want to start your voyage of discovery into algorithmic art by experimenting with fractals? If so, we suggest you take a look at *XaoS* and/or *Fraqtive*, which are free and available in the main repositories, and the screenshots (above-right and opposite, bottom-left), which give you an inkling of the sort of thing you could create. And if 2D fractals, like those produced by *XaoS* or *Fraqtive*, offer plenty of scope for experimentation, for the ultimate you might want to take a look at 3D fractals like the so-called Mandelbulb, as produced by the *Mandelbulber* software.

Evolutionary art

We're going to look at a form of generating artwork that's variously been called generative, genetic or evolutionary. It's based, to varying degrees of accuracy, on the processes involved in biological reproduction in which generations inherit characteristics from their parents, but usually with some random genetic mutations. Our first genetic art package is called *Evolvotron* (<http://timday.com/share/evolvotron/index.htm>), and it's freely available. According to its author, *Evolvotron* is interactive generative art software to evolve images, textures and patterns through an iterative process of random mutation and user-selection-driven evolution. It seems to be the evolutionary art package of choice right now, at least in the Linux community, although its implementation of genetics is somewhat basic.

When *Evolvotron* starts, it displays a grid of thumbnail images. These patterns are generated using various algorithms and their associated parameters, which, taken together, are the genetic information that define them, and are initially selected at random. If you don't like any of them you can tell *Evolvotron* to present you with a different palette of options. When you find one that's to your liking, click on it and a new selection of random images is offered, this time based on your selection, so much more focused than the initial palette. You can continue in this way until either you're satisfied with your creation or you get bored. When you find a good-looking image, you can save it as a graphic file and also as a data file that you can use to define a starting point when you next use



Evolvotron. After you've got to grips with the basics, there are lots of other options to play with that define, for example, the degree of randomness and just what, in more precise terms, is affected by mutations.

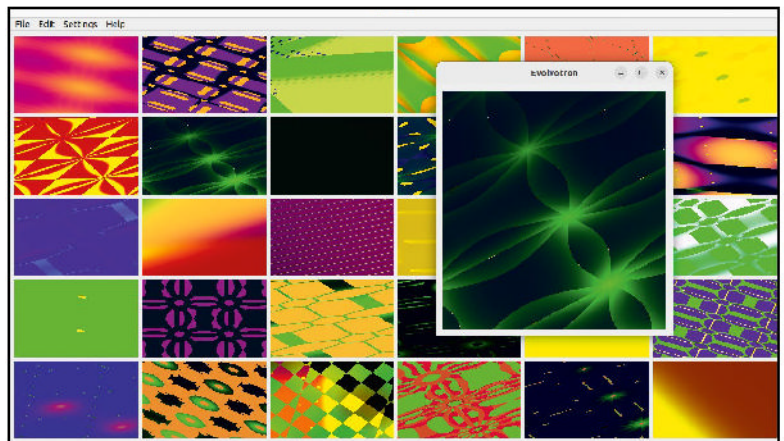
While *Evolvotron* offers an interesting approach to generating pseudo-random artwork, it doesn't really mimic the biological genetic process, in that it doesn't emulate sexual reproduction. In other words, each new image has just a single parent. To see a better example of breeding art from two parents, take a look at the Java application *Kandid*. Despite being somewhat long in the tooth, and one category of image no longer working due to the obsolescence of the necessary renderer, it's still quite impressive. Yes, it can work like *Evolvotron* in creating each generation from a single parent but, in permitting two parents' genetic information to be merged before applying mutations, it offers a lot more flexibility. It takes some time to really become proficient with *Kandid*, but that would be time well spent. In passing, you might be interested to know that much the same technique was used by artist William Latham, back in the '90s, to generate impressive rendered 3D images of, quite appropriately, biological-looking creations, albeit with a decidedly otherworldly look to them – see <https://mutatorvr.co.uk/ibm/>.

Latham has explained the benefit of the evolutionary approach by describing the process as like taking a navigated tour through an almost infinite gene space, and thereby discovering images that

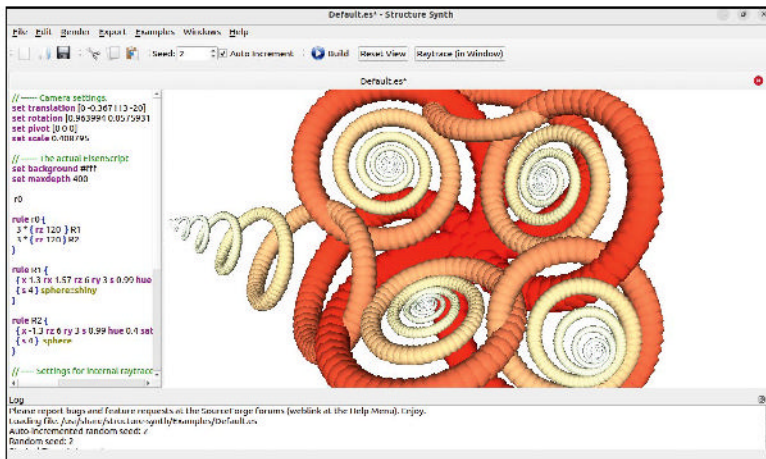
Going beyond simple zooms of escape-time fractals, plenty of mathematical options are available in many of the fractal packages, including XaoS.

QUICK TIP

Structure Synth generates three-dimensional objects by using a 3D printer, therefore, you can use this software to produce algorithmic sculptures.



To speed up the process of navigating an infinite number of random options, *Evolvotron*'s evolutionary approach enables the user to influence its evolutionary development.



This amazing 3D artwork was created in Structure Synth using the simplest of scripts, as shown in the panel to the left.

QUICK TIP

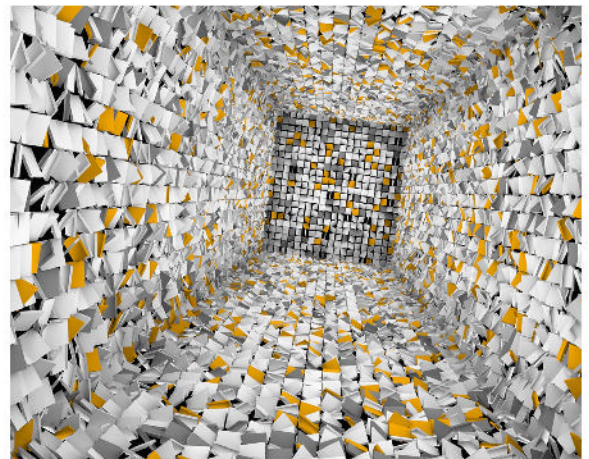
We might expect that visual art could be created from music, but most software creates little more than a colourful waveform display. However, the live coding Visor environment enables the coder to access the audio data to drive the visuals in step with the music.

would have remained forever hidden had he had just been presented with endlessly with random variations.

Structure Synth

While some fractal software offers you the option of defining your own formula, to investigate fractals other than the ubiquitous Mandelbrot set, the software we've seen so far is influenced mostly by entering parameters of various types. If you're itching to write some code, though – and we have to admit that the phrase “algorithmic art” does rather suggest that – there's plenty to keep you amused.

First in this category, we're going to introduce you to *Structure Synth* (*structure-synth* in the repositories, <https://structuresynth.sourceforge.net>), which offers you the possibility of creating some amazing rendered 3D objects. Admittedly, it doesn't allow you to do anything you couldn't do by hacking your own code from scratch. However, by providing built-in 3D primitives such as cubes and spheres, a renderer and an interface for manipulating the final 3D object on



In the hands of an experienced user, like Stinging Eyes, *Structure Synth* is capable of producing some spectacular 3D artwork.

screen, it means you don't have to re-invent the wheel and, instead, you can use your time and effort being creative. So, although you have to write some code – in the broadest sense of the word – that code, which is written in a specialist language called EisenScript, has the power to produce some complicated and impressive-looking results in just a few lines.

Although the EisenScript code can be as simple as adding a cube to the workspace, generally speaking you're going to be defining rules. A rule, for example, could define how an object is manipulated, by transforming its position, size, rotational angle and colour. When we further point out that rules can be called repeatedly, they can call each other, and they can be recursive, you start to get a feel for the power of this simple language. And since, in common with so many resources for algorithmic art, randomness is a key feature of *Structure Synth*, different versions of the same rule can be defined and, in this case, when that rule is called, one of the definitions is chosen at random, according to their defined weights. The seed of the random number generator can be defined so your creations are repeatable. And as a final comment, although *Structure Synth* has its own renderer, you can export a 3D object to a third-party ray-tracer.

» THE FIRST ALGORITHMIC ARTIST?

One of the first artists to be given the algorithmic tag was British artist Harold Cohen, who started his work on a CDC 3200 mainframe at the University of California, San Diego, in 1972, before moving to DEC PDP-11 and VAX mini-computers. But these were the early days of graphic output devices, and because pen plotters weren't too good at creating solid colour, Cohen often produced black line drawings, which he then coloured by hand. However, the main reason for mentioning him here is to illustrate more diversity in algorithmic art than we've seen in the main part of the article.

In particular, while his software, by definition, embodied logic, we'd be inclined not to think of it as overtly mathematical, in the sense of generating geometric patterns. Instead, Cohen hard-coded picture elements that could, for example, include parts of human figures, flowers and trees, and used the logic of his software to mimic the thought processes of human artists who were competent in reproducing the individual basic elements. In this respect, he was ahead of his time, because his work has been described as the intersection of computer artificial intelligence and art. Interestingly, while he first programmed his *AARON* software in C, he later used the venerable language Lisp, which has particular strengths in AI.

The Processing language

All the software we've seen so far – while providing almost infinite scope to produce something well and truly unique – is tailored to creating particular types of algorithmic art, often with their own characteristic look. To break out of this restriction, and find software that is limited only by your imagination, we need something fully programmable. In this sense, you could consider your favourite programming language – whether that's Python, C, Java or whatever – as the ultimate tool for algorithmic art. According to its adherents, however, *Processing* (<https://processing.org>) is the ideal programming tool for the algorithmic artist.

Processing is an open source Java application that provides a simple IDE for developing software in the language of the same name. The language, which dates back to 2001, is Java-like, it comes with a library of graphics primitives, and it's considered easy to use, an important requirement if you're targeting artists who want to take up the challenge and opportunities of

the algorithmic approach, but don't necessarily have any programming experience.

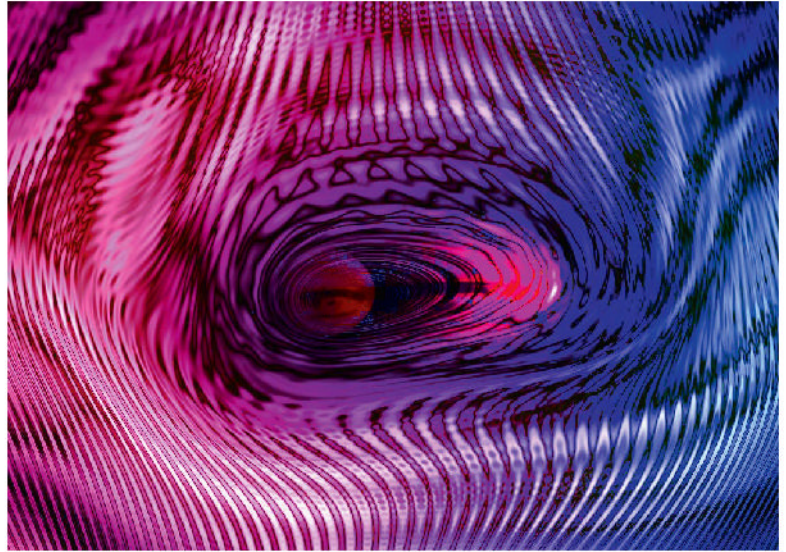
Processing isn't the only language with an extensive graphics library – indeed it would be hard to find a serious language that doesn't have good graphics support – and it's by no means the only language that is suitable for beginners to coding. This begs the question, therefore, of why you might choose Processing today instead of something more mainstream. We'd suggest that a key reason that it can be thought of as the artists' language is exactly because it's the artists' language. That might sound a nonsensical statement, so here are a few examples of why you might want to consider it. While introductory books on most programming languages don't make reference to algorithmic art, in the realm of Processing, you'd struggle to find any books or online courses that aren't tailored towards algorithmic art.

The Processing coder's first program, therefore, is far more likely to draw a square, therefore, than display 'Hello World'. Sample programs abound, of course, but pretty much all of that code has been contributed by artists to inspire other artists. And while it would be wrong to suggest that questions posted on *Processing* forums are all art-related – after all, even algorithmic artists sometimes need technical advice – the majority of people in these online communities are artists.

When you first start *Processing*, it would be a good idea to peruse some of the many example sketches (the *Processing*-ese word for program) that are provided with it, to get a feel for typical code. You'll soon discover that sketches can either be static or interactive. Static sketches are like the programs you'd write in most languages, in the sense that the instructions are executed sequentially, and the code eventually terminates unless it contains an infinite loop. These sketches generate static images. In the more commonly used interactive sketches, there are two main blocks: the **setup** block, which executes just the once, and the **draw** block, which executes repeatedly in rapid succession, thereby allowing dynamic artwork to be created – moving images, in other words. The following simple interactive sketch repeatedly draws randomly sized purple circles with no fill, centred at the position of the mouse pointer.

```
function setup() {
  createCanvas(800,600);
  background(100);
  stroke(255,0,128);
  noFill();
}
function draw() {
  circle(mouseX,mouseY,20 + random(50));
}
```

We'd be inclined to describe the result as barely art, but when we bear in mind that we've only added five lines of code to the usual template, the power of this approach is clear to see. And when we further remember that animation can be a lot more exciting than just following the mouse pointer around the screen, it's not surprising that so much *Processing* art is dynamic. What's more, it's a genre of the visual arts



Processing, the programming language for artists, really enables the creative juices to flow, as Edgar Ja demonstrates here.

that barely existed prior to computers becoming a part of the artists' toolbox.

As you might expect, to alter a *Processing* sketch, you have to stop its execution, edit the sketch, then try it out by executing it again. However, you might be interested to learn about the *Processing*-based *Visor* software, which allows you to modify the sketch as it's running and immediately see the result. In that respect it's like *Sonic Pi*, which allows music to be created and modified on the fly in a live coding event, which often involves the code being projected on-screen. *Visor* was developed to build on this to allow a 'code jockey' to enhance a musical performance – whether at an event involving a DJ, conventionally performed live music, or a live coded music performance – with live coded visuals to provide an audio visual experience. And there's the option of on-screen code to entertain the more technically minded among the audience. **LXF**

QUICK TIP

The Mandelbrot set is an example of an escape-time fractal, and we can think of it as an infinite pattern. Other fractals, however, exhibit self-similarity at all scales of magnification because the algorithm specifically defines that.



While this picture by Harold Cohen has every look of a human-generated abstract painting, it's actually algorithmically generated.

» DISCOVER MORE ALGORITHMS! Subscribe now at <http://bit.ly/LinuxFormat>

MAGIT

Credit: <https://magit.vc>

Making Git so easy, we say it's Magit!

Aaron Peters uses the Emacs Git client Magit to perform a long overdue update of his LXF repository.



OUR EXPERT

Aaron Peters is a long-time Linux user, but still quite the Emacs neophyte. Nothing a few decades' more experience won't fix. In the meantime, being a software designer will have to suffice.

Git was developed as a tool to manage the development of code by teams, including versioning and collaborating. In other words, a tool for developers. But its use is now so pervasive in the open source world that most serious users of Linux should be able to use it. While you may not write software, some applications' authors provide it for installation by cloning the repository. Additionally, participating in online projects may require you to make your contributions (see **LXF298** for a summary of non-code input) to a GitHub or GitLab repository.

But *Git* is a sizable application, with lots of subcommands and options. If you're simply a studious Linux user, rather than a developer or sysadmin, you may not be called upon to use it every day. The learning curve can therefore be pretty steep. This article will introduce **Magit**, an Emacs package that provides what some call the best GUI for *Git*. Let's get started.

Why Magit?

Magit (<https://magit.vc>) is a package for Emacs written in Lisp, so you need to install Emacs to use it. But why use this compared to some of the other GUI apps for *Git*? There's a couple of compelling reasons.

The primary reason is that it provides a complete interface to *Git*. Many *Git* clients offer a graphical way to do a limited number of actions. And while a GUI does

The package-install command makes installing Magit, including its dependencies, a breeze. You can also use it to keep it all up to date.

help with the learning curve, you may have to resort to the command line if you get stuck – and if you're new to *Git*, you will get stuck at some point. By familiarising yourself with **Magit**, you can take advantage of the GUI and have complete functionality at your fingertips.

Speaking of fingertips, like most things Emacs, **Magit** is largely focused on the keyboard. Once you get used to **Magit's** key commands, you'll be flying through your commits without the need to constantly reach for the mouse. The UI for **Magit** is also designed to evoke – but not reproduce exactly – the results of terminal commands. This means if you ever do need or want to use command-line *Git*, things will look very familiar. Of course, for existing or prospective users, it enables you to live your life in the delightful Emacs environment (see **LXF241** and **LXF260** for other swell things it can do).

Into the Emacs

To install **Magit**, you need to install Emacs. On DEB-based distributions, this should be a straightforward command to install the latest version:

```
$ sudo apt install emacs
```

For RPM distros, use the following:

```
$ sudo dnf install emacs
```

Once Emacs is installed, you need to get the **Magit** package. If you are or plan to be a moderate Emacs

QUICK TIP

If you're interested in **Magit** but not Emacs, add the following to the end of the `~/.emacs.d/init.el` file after installing to launch straight into Magit: (require 'magit) (magit-status)

Magit's status screen displays the repo's current state plus its recent commits, which can be expanded to show details.

user, getting a distro (or an advanced configuration for Emacs set up for easy installation) does not only get you **Magit**, but some clever customisations as well. Doom Emacs (<https://github.com/doomemacs>) does a great job of including and configuring **Magit**.

However the above also gets you a lot of non-Git cruft. If you're not planning on regular use of it for text editing, for example, recent versions of out-of-the-box Emacs come with the `package-install` command. This figures out any dependencies, fetches the required files from the MELPA (<https://melpa.org>) repository, and manages their installation for you. To install just this bare minimum, first add MELPA to your Emacs install by adding the following to the `init.el` file in your `~/emacs.d/` directory, or create it if it doesn't exist:

```
(require 'package)
(add-to-list 'package-archives
  '("melpa-stable" . "http://stable.melpa.org/
    packages/") t)
```

Launch/relaunch Emacs. Hit M+x on the keyboard. In Emacs-speak, M+<letter> is the notation for Alt+<letter>. Type `package-install`, then hit Enter. You see a prompt. Type `magit`, then Enter. If all is well, Emacs downloads and installs **Magit** from the MELPA repo.

Create, view and populate

Once installed, the first thing you'll probably do is set up a new repository. If you're installing something, you'll likely clone the software's repository; if you're using Git for your own project, you'll initialise one.

Cloning a repo:

Press M+x, type `magit-clone`, then press Enter. You're asked from where you'd like to clone it. We'll hit u for a URL. Let's clone the aforementioned Doom Emacs distro by pasting `https://github.com/doomemacs/doomemacs` into the mini-buffer, then hitting Enter.

You can clone the repo wherever you'd like, though if you want to use Doom, you can send it directly to your `~/emacs.d/` directory. **Magit** offers to set up the original repo as the origin for your default remote.

Creating a new repository:

Press M+x, type `magit-init`, then press Enter. You're prompted for the location for creating the repo. Note: **Magit** can't create bare repositories, so you need to

handle these from the command line. **Magit** opens that location in its status view as soon as it's created, although it will be somewhat boring, being new and all.

Practical Magit

At this point, we'll take up a truly worthwhile task: bringing the author's *Linux Format Git* repo up to date with the content of 10 unversioned articles. Press C+x, g on the keyboard (that's Ctrl+x, then g) to launch the main **Magit** interface, also available with `magit-status`. Doing this we can see the sad state of our *Linux Format* repository, with only Org-Mode and Flatpak articles (*are you asking for more work?—ED*) included as part of four commits. The top row displays info on the current commit of the repo, including the branch name and commit message.

Below this is a line that reads 'Recent commits'. You can navigate **Magit's** status screen with the arrow keys and use the Tab key to expand this list of commits. It is listed with the most recent first – in other words, what you see should match what's in the status line at the top. If you highlight one of the commits and hit the Enter key, you'll see the details of that commit in the bottom pane.

In Emacs terms, it opens a `magit-revision` buffer, which itself has two sections. At the top you'll find

The details of a commit list: which files were added, modified or deleted, as well as diffs for text.

QUICK TIP

A bare repository is a common way to designate the main repo for a Git project. This is a Git repo that only allows access to the project's version history. **Magit** doesn't provide a way to create these, so if you need one, you should create it via the terminal.

» KEY GIT TERMS

This article assumes some knowledge of basic Git terms. In case this is your first time digging into the workings of Git, beyond pasting a clone command into your terminal, here are some quick definitions.

- **Repository** A Git repository consists not only of all the files in your project, but also all the versions of them for which you've taken snapshots.
- **Commit** Each aforementioned snapshot of your project is called a commit. By creating commits, you can revisit (or even roll back) your project to

previous points in time – for example, to recover a deleted file.

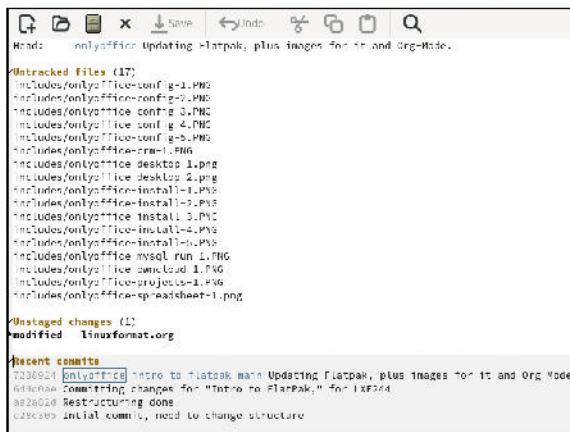
- **Branch** When you branch a project, you create a version of it with which you can experiment. In the event it doesn't work out, just delete the branch and try again. Other branches, including your main one, are unaffected.
- **Merge** If your fiddling in a branch is successful, you'll want to merge its changes back to your primary branch.
- **Remote** Git is a distributed system. You can configure remote repositories that you can send changes to, or collect

changes from, even if they are made by others.

- **Fetch** When you fetch from a remote, you are downloading metadata on what has occurred in that repository since the last time.
- **Pull** Pulling from the remote does the same as fetching, but also updates your project with the outstanding changes. It's basically the same as doing both fetch and merge.

Of course, there are many (many!) other Git commands and concepts. But the above will get you surprisingly far.

Be sure to refresh your Magit buffer while updating your project, and it starts queuing them up for commit.



the info on the commit, including who made it, the message and the branch. You'll also see whether the commit was part of a merge action. In the bottom section, **Magit** lists the contents of the commit, including new and updated files and, in the case of text files, the changes made.

Now we've gained some familiarity with the **Magit** interface, let's get some more serious work done.

Branch, commit, merge

Git's most basic workflow consists of branching off whatever is considered the project's primary one, making changes, then merging those changes back. **Magit's** status screen makes this easy. The first step when making changes to a project should be to create a new branch to capture your changes, or switch to a branch that already exists for this purpose. In order to

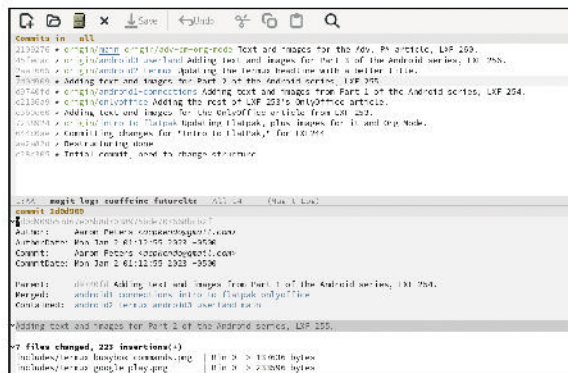
» MAGIT AND GIT RESOURCES

This article naturally doesn't cover the use of **Magit** in full. In particular, there are two other resources worth exploring once you're comfortable with basic concepts. The first is **Magit's** documentation (<https://magit.vc/manual/magit>), which talks about some of the more complex operations, such as handling merges and tagging. Another is **Forge** (<https://magit.vc/manual/forge>), a companion project that provides the same sort of interface as **Magit**, but for *Git* forges, such as GitHub and GitLab. Given how common these platforms are for collaborating on open source projects, it's worth getting to know this module, which is also installable from MELPA.

As noted at the start of this article, **Magit** isn't the only *Git* GUI available. Below are some other options available for Linux systems:

- **GitKraken** (www.gitkraken.com):
A very slick GUI for *Git* repositories. However, it's not free software.
- **Gitk**: Invoked with the `gitk` command from inside a repository, this basic repository viewer ships with *Git*.
- **LazyGit** (<https://github.com/jesseduffield/lazygit>):
A text UI client that provides a panel layout and single-key commands similar to **Magit**.
- **Fugitive** (<https://github.com/tpope/vim-fugitive>):
If you're one of those people who prefer *Vim*, *Fugitive* is a plugin for *Git*, although it doesn't provide a UI.
- **Guitar** (<https://github.com/soramimi/Guitar>):
A nice-looking FOSS *Git* client written in *Qt*.

Other options for interacting with *Git* include plugins for file managers, text editors or development tools. Of course, you could always just learn the terminal commands as well...



After adding remote repos, your Magit log views display your interactions with them as part of the history.

create a new branch, press b (**Magit's** mnemonic key for Branch), then n (for New), from the status view. Notice when you press b, *Emacs* displays other command options for you in the mini-buffer. Alternatively, you can also press M+x, then issue the `magit-branch-create` command. First, select the commit from which you want to create the branch. This defaults to either the most recent one or a specific commit if you have one highlighted within the expanded Recent Commits list. Once you select your starting point, you just need to provide a name for the new branch at the prompt, then press Enter.

Next, check out the new branch so you can start making your changes. Like above, press `b`, then `c` for Checkout. A prompt appears asking you for the name of the branch to check out. One helpful tip is that hitting the `Tab` key helps you enter valid values for your current action. If you press `Tab` at a blank prompt, it lists your options – in this case, all existing branches. If you enter some text and press `Tab`, it attempts to autocomplete using those branches.

Next we'll make some changes to the project. With the status screen open, start adding, modifying or deleting some of the files. In this case, we are adding the text and images from an article on deploying *OnlyOffice*, all the way back from **LXF253**. Pressing the g key in the **Magit** window refreshes the view, which clearly shows the new (untracked) and updated files. Again, using the arrow keys to navigate, you can visit each file and press the s key to stage it for a commit. The u key (for Unstage) removes them, while the uppercase versions S and U stage/unstage all files, respectively. Note how **Magit's** Staged Changes section populates with the files you include or exclude using these keys.

Once you're ready to make the commit, press `c`, then `c` again. All the files in the Staged Changes section are included in a new buffer, where you can type your commit message. When finished, hit the `C+c`, `C+c` key combination to save the message and finalise the commit.

The final step in this basic workflow is to then merge the changes from the working branch back into the primary one. Again, from the **Magit** status screen, hit b, then b to do a checkout, and select your main branch (probably named **master**, or more recently, **main**). Then press m, then m again to start the merge process. The prompt asks for the branch you want to merge in, so select the one you just created for those

recent changes and press Enter. If all goes well, you should see the merge info pop up as an update to your most recent commit.

View repository history

As your repository grows, there will come a time when you have to inspect that history. It might be to understand when a mistake was introduced (though surely not), or to go back and grab some content that was accidentally deleted. **Magit** provides useful commands to browse the repo's history, enabling you to fix whatever needs fixing.

From the **Magit** status screen, start with the **l** key (for Log). Like other functions, you're presented with the options available from this leader key, including:

Key for log	Function
s	Selects the current row
u	Displays the log for the current branch and its remote counterparts
a	Displays the log for all branches

The log displays a list of commits based on your selected parameters, including the commit ID, message, committer name and how long ago the commit was made. Similar to the list of recent commits on the main status screen, you can select one of the commits in the list with the Enter or Space keys. The **Magit** revision buffer that appears displays the info about the commit.

Working with remotes

While the above will equip many users to keep nicely versioned local repositories, it's more common to share your work with others. **Magit** provides the same sort of simple tools to set up your remote repos, push your changes to them, and fetch subsequent updates from them. First, use the commands with the **M** prefix (for reMote, because **r** is for Rebase) to get started.

The **magit-remote-add** command creates a remote for you, which you can issue by pressing **M**, then **a**. The first prompt asks for a name for the remote. You can name this anything you want, although a common convention is **origin** if it's the only one (even if you didn't clone it from there). **Magit** then confirms if you want to set this new remote repo as the default for both push (the **remote.PushDefault** variable) and fetch (**remote.FetchDefault**) actions. Note that if you point your local repo at a remote one that's bare, you may see an error that **Magit** can't find branches or other refs. Don't worry, we'll clear that up in a minute.

When your repo is configured, push your changes by pressing **P**, then **p**. This pushes your current branch to the remote. However, it's likely you'll want a full history of your project, so you have to push other branches one by one. Use **P**, then **o** (for Other Branch), and select the name of each one.

The final leg of this collaborative workflow is to get changes made by others from the remote repo. Keeping with its mnemonic pattern, **Magit** uses the **f** key to lead commands related to Fetching. Unless

you're doing something specific where you want to target what you get, pressing **f** and then **p** fetches from your configured remote – usually **origin**. This shows you any commits you haven't pulled yet, which you can do with the **F** key, followed by **p** for your configured remote.

The goal of this article, like others regarding *Emacs*, is to illustrate that with knowledge of a fairly small number of commands, you can still get work done quickly. **Magit's** mnemonic patterns make it easy to remember the key commands, and with the text-based UI, you don't need to memorise what can be very involved terminal commands. The following table summarises the main commands you'd need to complete a basic workflow:

Command	Keyboard	Description
magit-status	C+x, g	Displays the main Magit status screen
magit-clone	C, C	Clones a <i>Git</i> repo from the URL or path provided
magit-init	l	Initialises a new, blank <i>Git</i> repo
magit-branch-create	b, n	Creates a new <i>Git</i> branch from the current (or provided) commit
magit-checkout	b, b	Checks out the branch provided at the prompt
magit-stage	s	Stages the highlighted file for a commit
magit-unstage	u	Removes the highlighted file from staging
magit-commit-create	c, c	Creates a new commit from staged updates and displays a message buffer
magit-remote-add	M, a	Prompts for the name and location of a new remote
magit-push-current-to-pushremote	P, p	Pushes the current branch to its configured remote
magit-fetch	f, f	Fetches info on updates from the current branch's configured remote
magit-pull	F, F	Incorporates changes from the configured remote into the current branch

So, if you like *Emacs*, or are just looking for an easy and useful client program for *Git*, give **Magit** a try. 

QUICK TIP

Automate **Magit** refresh with this in your **init.el** file:
(add-hook 'after-save-hook 'magit-after-save-refresh-status t)
Note that this only works when you modify files from within *Emacs*.

» **WHY DO WE CALL SUBS GITS...** Subscribe now at <http://bit.ly/LinuxFormat>

Deck the hardware

We revisit and trick out the Valve Steam Deck a year on from release.

The year 2022 will go down as the year of the Linux Deck-top. A typo in *Linux Format*, you cheer, but for once, 'tis deliberate. Valve released its Linux-powered Steam Deck, driven by an abundance of cutting-edge open source technology that at least in part was backed and developed by Valve itself.

Back at the start of 2022, we covered the pre-release build-up and launch of the Steam Deck – see our preview and review in LXF287 and LXF288, and our getting started guide in LXF289. While some reviewers had misgivings about a Linux-powered handheld gaming device, the world didn't. A year on, the Steam Deck is a smash hit. Valve can't make them fast enough and it's the biggest grossing

product on Steam – with new games costing £60, that's a bigger deal than you might at first think.

So, here we are, a year on, a few Hardware of the Year awards under the Steam Deck's belt, along with plenty of software updates – see news (page 6) for how Valve is funding connected open source projects – and we're taking time to look at the Steam Deck once again to see what the latest developments are, what you can do with it and how its performance can be tweaked. One of the important improvements over its launch is that you can actually buy the Steam Deck now. Originally, Valve was having issues fulfilling pre-orders – due to the post-pandemic chip shortage – but it seems orders are now being sent out within two weeks.



If you're considering buying a Steam Deck, bear in mind that storage is really easily expanded with its SD slot. This makes the less-attractive 64GB base model very appealing, as you can trick it out with a reasonably fast 256GB microSD card for as little as £18 or 128GB for just £11, and this can be big-name cards such as Samsung. The sudden drop (at the start of 2023) is due to the introduction of 1TB microSD cards pushing down prices of what you might have previously considered large capacity models. The Steam Deck currently has a 2TB maximum microSD card capacity, but that shouldn't be an issue right now. Running from an SD card is slower than internal options, but our testing of the NMVe model put loading Valve's Help Desk after a cold reboot at 19 seconds from SD card and 12 seconds from the NMVe.

During testing, Valve changed how the Deck handles new SD cards – the single biggest change was moving to a quick format – but it still says it's "safest" to power down the Deck before adding or removing an SD card. They are hot-swappable, but just to make sure it's not in use, it's best to shut the Deck down.

We'll look at more ways of upgrading the storage below, but the speed of these larger capacity microSD cards is fast enough that you won't be inconvenienced waiting for them to load or invalidate any warranty you have left. There is a third way to expand the storage via the USB-C port, but it's the most inconvenient option.

If you're eyeing up a microSD card bargain, ensure you pick one that's still as fast as possible. Advertised speeds can top 140MB/s and are usually a healthy 100MB/s. The old write Speed Class (shown with a large C) topped out at Class 10 or 10MB/s. As this is regularly exceeded and, in fact, different use cases are better served by different abilities (read speed, write



speed, random access), there are now three new speed markings you should look for.

UHS: This is confusing as it's actually used to denote two standards – the bus type and the minimum speed supported. The Steam Deck has a UHS-I bus, so there's no point spending more on a UHS-II microSD card. The theoretical maximum transfer speeds for UHS-I is 104MB/s; for UHS-II this is 312MB/s. The second part of the rating is denoted with a U symbol and a number 1, 2 or 3 inside, respectively marking a minimum 10MB/s, 20MB/s and 30MB/s write speed.

Video Class: Marked with a V and a number that denotes the MB/s, starting at V30, V60, then V90, this relates to 30MB/s to 90MB/s minimum write speeds.

App Class: The latest measure switches to IOPS (input/output operations per second) and is denoted by A1 and A2. It is supposed to guarantee a minimum 1,500r/500w IOPS for A1 and 4,000r/2,000w IOPS for A2. Some tests show that A1/2 seems to be least useful in gauging an SD card's performance, while for loading games it's less useful than straight read speed.

We can report good results from the Samsung Evo Select 256GB microSDXC and generally recommend

The Steam Deck has become renowned as an awesome retro gaming platform.

Samsung SD cards are currently coming on and off offer; a 512GB high-speed card like this can go for £50.



» UPGRADE YOUR SSD

Another user-friendly point Valve has pushed about the Deck is that the company hasn't done anything to prevent it being opened, repaired or upgraded. Now, that's not to say you should go around opening, repairing or upgrading it yourself, but the lovely types at [iFixIt.com](https://www.ifixit.com) rank the Deck with a 7/10 for repairability. Plus points are its modularity, single screw size use and its ease of opening, while the main negatives are that the

battery and charge port are tricky to replace.

However, this means that for someone capable of upgrading the built-in SSD – especially for the cheaper and smallest 64GB model – it can be done relatively easily and provide copious more high-speed internal storage. It would be tempting to buy the biggest and fastest SSD, but do not do this; the 64GB Steam Deck uses a single-sided M.2 2230 Gen 2 x1, while the larger capacity

models use an unusual NVMe M.2 2230 Gen 3 x4. Sabrent has a 256GB NVMe M.2 2230 for around £50 as an example, but bear in mind that the same money would buy a 512GB microSD.

We're not going to go into replacing an existing Deck SSD here – that's an entire tutorial in itself and we wouldn't do it while you have a valid warranty, either. As it is, iFixIt maintains a detailed SSD replacement guide, alongside instructions for

almost every component on the Steam Deck.

SteamOS recovery to the new SSD is pretty standard stuff. Valve has a recovery guide at <https://bit.ly/lxf299image> and it isn't much more complicated than downloading the latest recovery image, writing it to an 8GB (minimum) USB drive or microSD and booting that via the Boot Manager that's accessed by powering on while holding down the Volume Up button.



A USB-C dock of any description can transform the Steam Deck into a desktop Linux system.

looking for a UHS-I U3 V30 microSDXC card. Due to the hardware limits on the Steam Deck, don't pay extra for cards with read speeds higher than 100MB/s or write speeds higher than 60MB/s.

It's just a PC!

Before its release, Valve was vocal (and still is) about the fact the Steam Deck is "just a PC". From its point of view, that means you can install any software on the Deck you like, but it also means it's using commodity hardware and standard protocols, too, so you can expand or upgrade the Deck if the components allow.

Towards the end of 2022, Valve released the official Steam Deck Dock (try saying that three times quickly). This plugs into the USB-C connector and provides a host of ports: USB 3.0, HDMI 2.0, DisplayPort 1.4, power and Ethernet networking. This effectively transforms the Deck into low-end Linux desktop. While the idea is you're able to more easily game on a large display via wireless gamepads, you can add a keyboard, mouse and even storage over the faster USB-C bus.

At £79 a pop, the official Valve Deck Dock is pricey for what is basically a USB-C dock. If you peruse bay-like sites or rainforest-inspired retailers, you'll find near-identical docks for less than half the price – do check the reviews before buying and ensure it supports 4K @ 60Hz; the cheapest options likely only do 4K @ 30Hz.

If you want, you could plug a USB-C capable display directly into the USB-C port. More sensibly, as we've

suggested, is using one via a dock. In this case, the Big Screen SteamOS display appears on your external monitor. On a TV, you may need to tweak the Display Scaling using the Steam Button > Display Menu. There are resolution settings and scaling controls.

To use the Steam Deck as a Linux desktop, select Steam > Power > Desktop (this doesn't appear if you're in Family Mode). The default setup is for dual-display, with the external monitor active as the secondary display. It's likely you'll want to reverse this, so right-click a blank desktop area and select Configure Display Settings. Select the external display, then Primary > Apply. You may want to clear Enable on the internal display to disable that too.

If you do fancy using your Deck more as a PC, and so want to make it boot directly to the desktop, open the terminal and type:

```
$ steamos-session-select plasma-wayland-persistent
```

Previous versions of SteamOS left out the Wayland part, but the latest updates require it. Do keep in mind that SteamOS isn't designed or maintained to be a desktop-first operating system – this isn't the year of the Linux desktop – so we found some oddities during use with the touch controls and the onscreen keyboard not responding. However, the whole point of this exercise is to be using a keyboard and mouse...

If you find you're not getting on with the Desktop Mode, you can easily switch the login back to the SteamOS Mode with the following command and a quick reboot:

```
$ steamos-session-select gamescope
```

Hands on Deck

Valve has created a sensational gaming experience out of the box – it's brought a console-like experience to the PC using that Linux thing we all like to complain about! That's not to say it's perfect, though. Valve has been rolling out updates throughout 2022 and is continuing to do so, with non-trivial HDR support, for example, being lined up in 2023.

Away from official Valve-created updates, the simplest way to tweak and extend SteamOS is to use a plugin system. We've run through Decky Loader in the boxout (see below) but there is also Crankshaft, which seems just as slick but is apparently less popular. Let's

» GET DECKYED OUT

Let's face it, we're not using Linux because we want an easy life, but because we like to tinker, play and extend things. As SteamOS is basically Arch Linux running on an immutable filesystem, you can tinker to the degree that Flatpak and SteamOS enable it. One system you absolutely must try is Decky Loader. This is a widely supported plugin system for SteamOS that offers a host of cool and handy plugins –

find it at <https://github.com/SteamDeckHomebrew/decky-loader>.

Be aware that Decky Loader is incompatible with Crankshaft, so you have to remove that first. It's another excellent plugin system that uses ports 1337 and 8080, so anything using these needs to be removed or changed, such as *Syncthing*.

Before doing anything, let's enable Developer Mode. Press the Steam button and

scroll down to Settings > System > System Settings > Enable Developer Mode. Scroll down to Developer > Miscellaneous > Enable CEF Remote Debugging. It's worth rebooting at this stage.

Press the Steam Button > Power > Switch To Desktop and wait for the desktop to load. At this point, you can open a web browser, navigate to the GitHub above and use the Download button to grab the installer and run

it. If you don't have a password set, it uses Decky! as a temporary one.

There's a terminal install option using this command for the pre-release version: `$ curl -L https://github.com/SteamDeckHomebrew/decky-loader/raw/main/dist/install-prerelease.sh | sh`

Return to Gaming Mode, open the Quick Menu, locate the new Plugin icon and then use the Shop icon to install Decky plugins.

take a quick look at the essential plugins you should be running.

SteamGridDB

A huge issue with the otherwise slick SteamOS interface is the default gaming artwork – the game covers and ‘hero’ banner images that appear in the background when you click on a title. They’re fixed and can’t be changed, while non-Steam titles just appear as blank rectangles. Enter SteamGridDB. Get it installed and when you select a game, click the Settings cog on the far-right and Change Artwork (on the main SteamOS list, select a game and use the menu button, too). This opens a wide selection of artwork options.

Autoflatpaks

We’re all about laziness here at *Linux Format* Towers and this plugin cuts the need to manage any Flatpak applications you might install for use with the Linux desktop. Once installed, you can select an update interval of days or hours, plus there are options to check at startup and install updates unattended.

ProtonDB

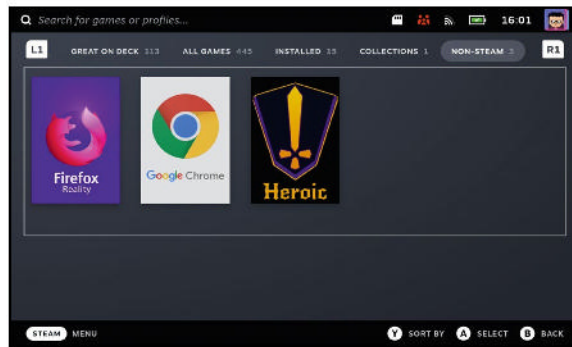
Wine and, more recently, ProtonDB offers ratings on how well Windows games run, typically with a Platinum, Gold, Silver or Bronze badge, but there are user comments, too. This plugin sucks in these ratings and displays them on the game page with a link to the dedicated ProtonDB page for complete details.

Steam is vapourware

The Steam Deck has access to hundreds of games through Valve’s own storefront, but there are even more ways to get titles on to the portable powerhouse. It’s remarkably easy to add game libraries from GOG (Good Old Games) and the Epic Games Store through an application called the *Heroic Games Launcher*.

Both stores are present on *Heroic*, but this same process works with more launchers, too, including the likes of Battle.net and Ubisoft Connect – although your mileage varies based on how well optimised the games you play are for the Steam Deck (if at all). Bear in mind that just as with Steam’s catalogue, not all games will run – *Fortnite*, for instance, is a big name that’s missing.

Heroic is a wrapper that enables the Epic Games Store and GOG store to run on Steam Deck, and it can open titles from both, where compatibility allows. It has



Get all-new fancy artwork for existing Steam and non-Steam titles.

its limitations – the actual storefront of each is essentially a browser window, for one – but it offers the same compatibility options through Proton as Steam.

Switch to the desktop – press the Steam Button > Power > Desktop. Click the shopping bag icon in the bottom-left. This opens the Discover store, where you can search for *Heroic Games Launcher* in the search bar. Click Install, wait for it to finish, then enter the store page. Click Launch in the top-right corner.

Now you need to add the *Heroic Games Launcher* to Steam. While you can open the launcher in Desktop

EXPANSIVE TASTE

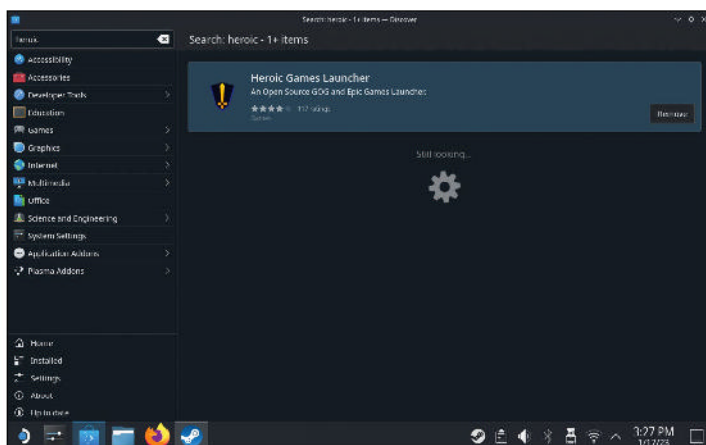
“Before its release, Valve was vocal about the fact the Steam Deck is ‘just a PC’. That means you can expand or upgrade it if the components allow.”

Mode, we recommend making it easily accessible on the main Steam interface first. To do this, open Steam (within Desktop Mode) and click Add A Game in the bottom-left corner, then click Add A Non-Steam Game. Now select *Heroic Games Launcher*, then close Steam and double-click the Return To Gaming Mode icon on the desktop.

Now that you’re back in the Steam interface, head into your Library and move to the Non-Steam tab. Here, you’ll find *Heroic Games Launcher*, which you can now open – don’t forget to change the artwork if you’ve installed SteamGridDB, mentioned above.

You’re asked to add login information for both GOG and EGS here, so be sure to have your passwords handy. Your library should auto-populate, and if you regularly take advantage of the free games included on both stores, you may be surprised by how much you have.

From here, it’s simply a case of installing the titles you want to play, before clicking the green Play button. If you do have compatibility issues with any game, clicking the yellow cog underneath the library icon enables you to pick different versions of Proton, which may help fix audio or visual issues. From this menu, you can also enable Esync and Fsync, too. **LF**



You can get more than just Steam games on the Steam Deck...

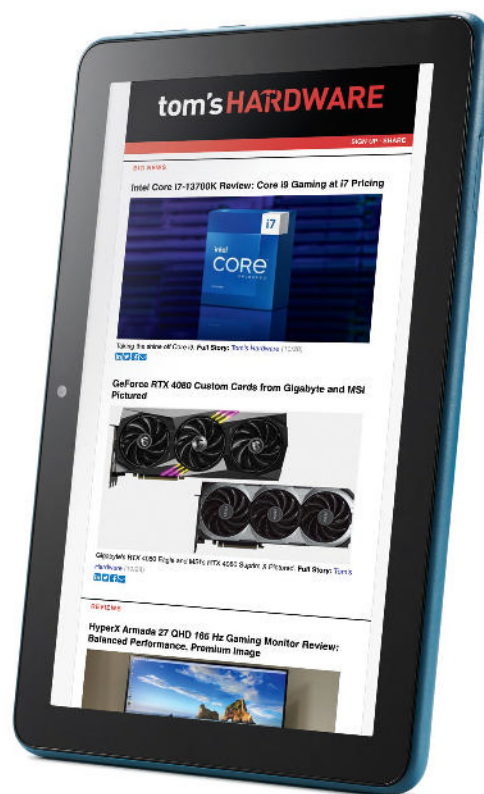
tom's**HARDWARE**

**GET ALL THE ESSENTIAL BREAKING
NEWS FOR THE TECH ENTHUSIAST!**

No matter if you're building a PC,
buying a laptop or learning about
robots, Tom's Hardware has all the
comprehensive knowledge you need.



**Scan & Subscribe
for free!**



The home of technology
techradar.com

HotPicks

Fotoxx » Saber » Upscayl » LibreWolf » Mprocs
 » Junction » Myuzi » Libre TrainSim » Infinite
 Shooter » OTPClient » HydraPaper



Mayank Sharma

found solace watching the World Cup and compiling *HotPicks* as he struggled to fight off a nasty infection his bug magnet brought home from school.

IMAGE EDITOR

Fotoxx

Version: 22.50 **Web:** <https://kornelix.net/fotoxx/fotoxx.html>

Image manipulation apps come in all shapes and sizes, but their list of features is usually inversely proportional to their usability.

Fotoxx is a good exception. It offers a rich set of retouch and edit functions that go beyond merely changing brightness, contrast and colour, and yet it isn't all that cumbersome to navigate.

The app has comprehensive image management functions and can, for instance, help you sort through your gallery using the metadata in the images. It can also import and edit RAW images, which makes *Fotoxx* suitable for professional photographers as well.

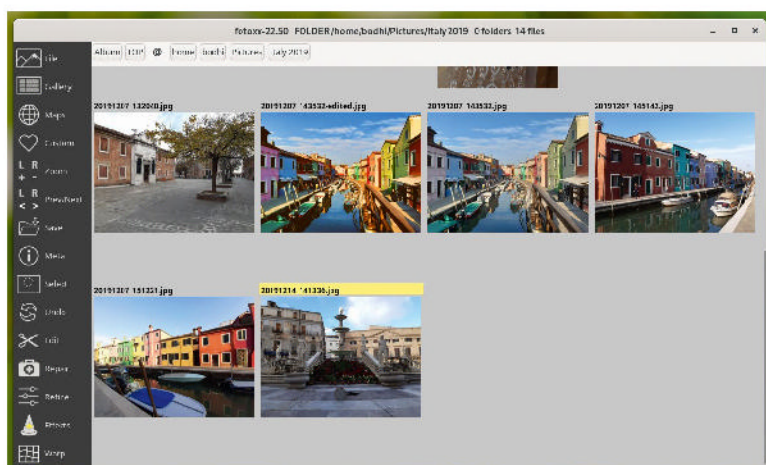
Like most dedicated image editors, *Fotoxx* enables you to select an object or area within an image using various tools such as freehand outline, follow edges, select matching tones, and so on. There is a special tool to flatten photos of curved surfaces, such as a page from a thick book or a poster on a round column.

You can also use the app to reduce noise and remove dust spots, and create collages, mashups and slideshows with animations. There are also several artistic effects to help convert a photo into a line drawing, sketch, painting, embossing, cartoon, dot image or mosaic. You can also use it to create panoramas by stitching several photos together, while the app ensures the brightness and colour remain constant through all images.

Not all distros carry *Fotoxx* in their official repos and many that do often have an older version. Your best bet is to grab the official Applmage from the project's website and make it executable, either using the file manager and heading to properties, or with the **chmod +x** command.

When you fire up the app, it makes sure you have all the required tools and libraries it needs. If there's anything missing, *Fotoxx* asks you to grab them from the distro's official repos.

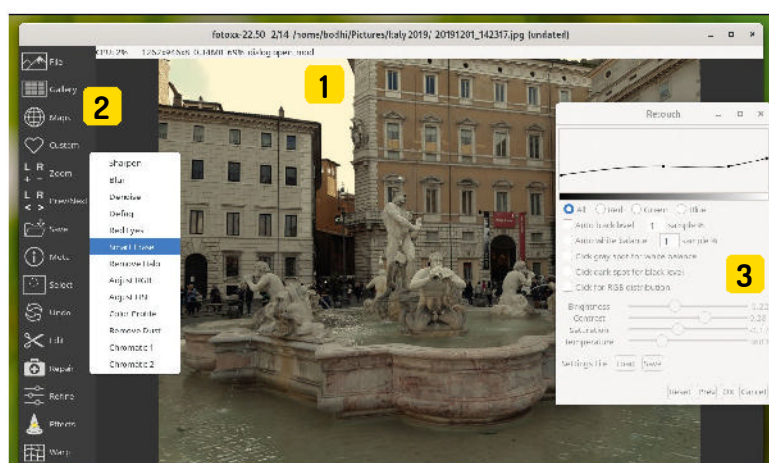
On its initial launch, *Fotoxx* brings up a dialog box to index your image library. This process can take some



You can use *Fotoxx* on resource-strapped machines and the app can be navigated entirely with the keyboard.

time, depending on the number of images you have in your library. While not unintuitive, we'll concede *Fotoxx* has an esoteric interface, which is why it's best to read through the documentation on the project's website to get a feel for the app.

LET'S EXPLORE FOTOXX



1 Main window

The image currently being worked upon is displayed in the main window. You can select the Gallery option from the left-hand panel to display a grid of all the images you have stored in the library, as shown in the image at the top of the page.

2 Toolbar

All of the app's features can be accessed from the left-hand side panel, where they are clearly and logically labelled.

3 Tool settings

Most of the tools will display an additional settings dialog box with relevant options here.

HANDWRITTEN NOTES

Saber

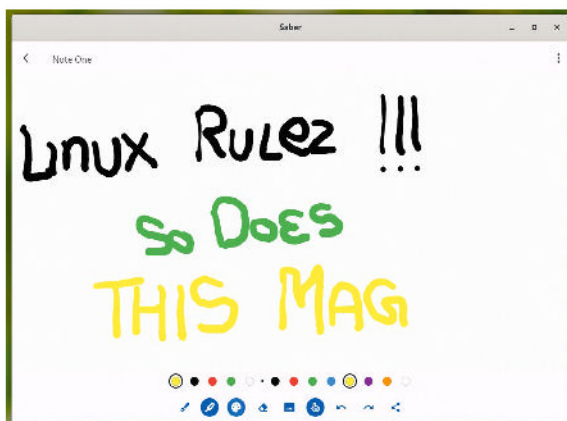
Version: 0.4.3 Web: <https://github.com/adil192/saber>

Saber still identifies itself as a work in progress, but we found the cross-platform app to be very usable. You can install it from Flathub with `flatpak --user install com.adilhanney.saber`.

The app has a simple interface. Click the + icon to create a new note. The fountain pen is the default writing tool, and there's also a highlighter. For both tools, you can select from one of eight listed colours, the default being black.

You can also swap the fountain pen for a ballpoint if you prefer, and set its size anywhere between 1 and 25, with the default being 10. Similarly, for the marker, you can choose a size between 10 and 100, in increments of 10, with 50 being the default. Besides these, in terms of tools there's also an eraser and an image uploader, which pastes an image from your computer into the note.

As for writing surfaces, in addition to the plain canvas, the app offers three other page types: a lined page with a margin, another without, and a third page with square boxes, much like graph paper. While



Head to <https://saber.adilhanney.org> to play around with a hosted version of the app.

selecting the type of page, you can also change the height of the line.

The app works best on a touchscreen device, such as a phone or a tablet, but can also be used effectively with a mouse. For the fountain pen, the speed with which you move the cursor defines the thickness of the line, much like a real pen. This doesn't work on the ballpoint pen, which has a constant thickness.

If you hold down the pen for a couple of seconds, the app draws a straight line between the starting point and the ending point.

When you've finished, you can export your note as a PDF file. There's also an option to export it as PNG, but that was always greyed out in the release we tested, probably because the software isn't feature-complete just yet.

IMAGE UPSCALER

Upscayl

Version: 1.5.5

Web: <https://upscayl.github.io>

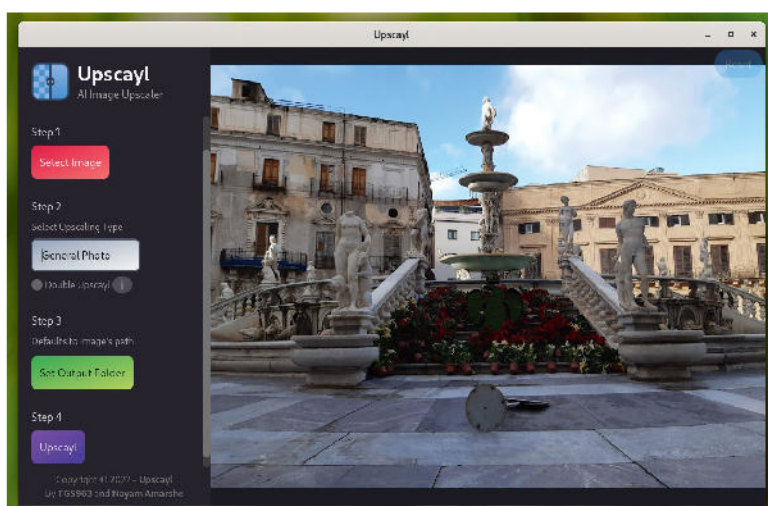
Upscayl is a wonderful little app that calls on advanced AI models to upscale and sharpen low-res images into ultra-HD resolutions.

The app works with images in JPG, PNG and WebP formats. In addition to working with individual images, it can also batch process multiple images at one go.

Upscayl is available as an AppImage for Linux users, so grab the latest release from the project website and make it executable, either using the file manager and heading to properties, or with `chmod +x`.

The app has a simple interface, which divides the image upscaling task into several easy-to-follow steps. The first is obviously to point the app to the low-res image. If you toggle the Batch Upscale option, the app enables you to point to a folder instead of a single image. Make sure the folder doesn't contain anything except images in the supported formats.

The next step involves selecting the upscaling type. The default is General Photo, which is what most of us will want to use a lot of the time. Besides this, the app also has an option to upscale digital arts. There's also



the Sharpen Image option, which you should use to bring out more detail in blurry images.

Finally, point the app to where it should place the upscaled images. By default, it drops them in the same folder as the source images. Hit the Upscale button and let it rip. There's no warning, but the process is compute-intensive, and freezes your computer. It can also take some time, depending on the specs of your computer and the number of images you've asked the app to upscale.

Upscayl didn't disappoint us with any image we threw at it, and has given us the confidence to handle the camera at parties even after getting a little tipsy.

You can use the slider to compare the original against the upscaled image after the app has done its magic.

WEB BROWSER

LibreWolf

Version: 108

Web: <https://librewolf.net>

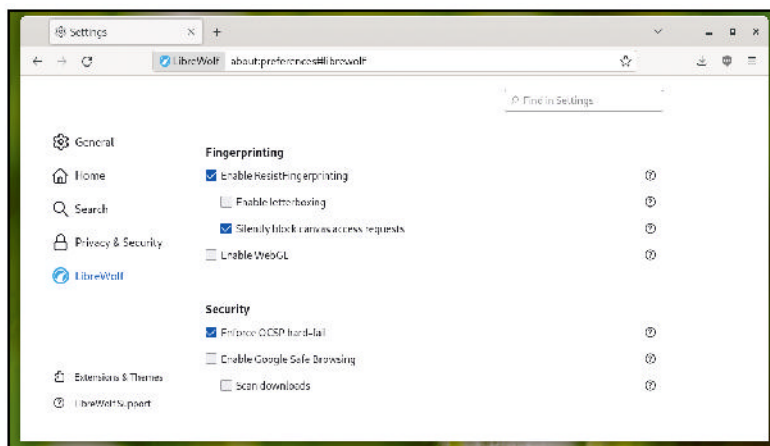
If you love *Firefox* and are really concerned about your privacy on the internet, but don't want to spend time equipping your browser with the relevant extensions, *LibreWolf* is for you.

The app hosts repos for various distros, but can also be installed from Flathub with `flatpak install flathub io.gitlab.librewolf-community`.

It looks pretty much like *Firefox*, since it's just a customised version of *Firefox* designed to increase protection against popular tracking and fingerprinting techniques. It does this by making certain under-the-bonnet tweaks to *Firefox*. For instance, it zaps all the telemetry and data collection procedures, and disables features that can be abused, such as search and form history, form autofill, *Firefox* sync, disk cache and such.

The app also clears temporary files on close, and enables HTTPS-only mode. It also always forces user interaction when deciding a file's download location.

LibreWolf defaults to using DuckDuckGo, but also gives you the option to replace this with another privacy-respecting search engine, such as SearXNG,



MetaGer and more. Furthermore, it includes the popular uBlockOrigin (uBO) extension for content filtering with custom default filter lists, and tracking protection set to strict mode to block trackers and ads.

To enhance privacy, *LibreWolf* strips tracking elements from URLs, both natively and through uBO, and enables the dFPI feature, popularly known as Total Cookie Protection. It also enables RFP, which is part of the Tor Uplift project and is considered one of the best anti-fingerprinting solutions.

To that end, it also prevents access to the location services of the OS, and uses Mozilla's location API instead of Google's. It also always displays the user language as en-US to websites, in order to camouflage the language used in the browser and in the OS.

One of the best aspects of the browser is its customised settings interface that exposes the most important privacy and security settings.

PROCESS MANAGER

Mprocs

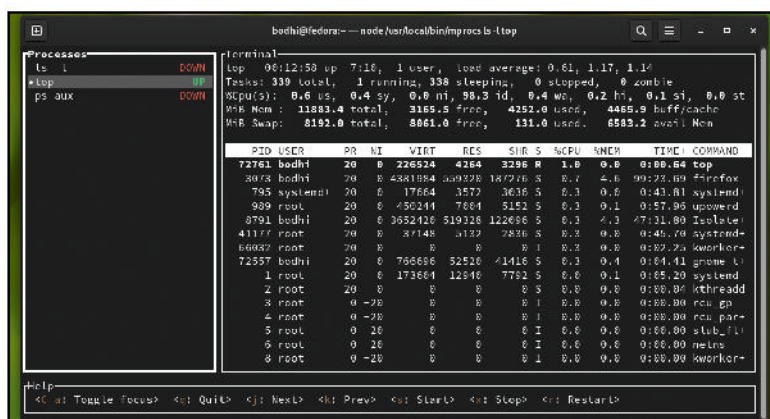
Version: 0.6.4 Web: <https://github.com/pvolok/mprocs>

If you've ever wanted to run multiple Linux shell commands at the same time, you've probably used either *Tmux* or *GNU Screen*. These tools, however, are usually used to run much more long-lived processes. If you have a bunch of commands that you run in parallel, regularly, look at *Mprocs*, which also displays the output of each command separately.

Mprocs is written in Nodejs, so to install it, first grab the Nodejs package manager (*npm*) using your distro's package manager. For instance, on Fedora use `sudo dnf install npm`, or install it on Ubuntu with `sudo apt install npm`. When it's done, install *Mprocs* with `sudo npm install -g mprocs`.

To use *Mprocs*, simply run the tool along with a list of commands you want to run simultaneously. Commands that have multiple options should be placed in quotes. For instance, `mprocs top "ls -lh"` brings up *Mprocs* with the two specified running in parallel.

The *Mprocs* interface lists the commands as tabs on the left-hand side of the screen, and on the right you have the output of the selected command. You can use



The words UP and DOWN indicate whether the command is still running, while the s and x keys can be used to start and stop processes.

the cursor to switch between any of the listed commands. You can use the Ctrl+A key combination to switch focus between the list of processes on the left and the output of the currently running shell process on the right.

To add a process to an ongoing *Mprocs* session, press the a key and then enter the command to launch a new process in the current *Mprocs* session.

If you know which commands you need to run regularly, you can drop them inside a file dubbed `mprocs.yaml`. The advantage of using the `mprocs.yaml` file is that it enables you to add aliases or nicknames to the commands for easier recognition within the *Mprocs* interface.

APPLICATION SWITCHER

Junction

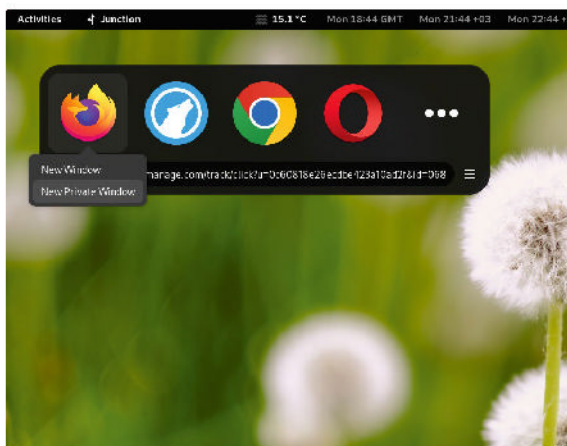
Version: 1.6 Web: <https://github.com/sonnyp/Junction>

Do you need to use different apps to open the same type of file depending on what you want to do? For instance, while you usually want to open an image in an image viewer, sometimes you want to send it to an image editor instead. In such a case, the Open With context menu option comes in really handy. But if you need to make this choice a lot, bringing up the Open With dialog box can become exhausting.

Junction is a nifty little app that makes it a little easier to choose the correct app to open a file. With *Junction* installed, simply set it as the default app for opening a particular file type. When you now double-click to open the file, *Junction* pops up and displays all the relevant apps that can handle the file.

While it does add an extra step in the file opening process, it can be a really handy option for files that you need to regularly shuttle between multiple apps. Also, you can use this extra step to modify URLs before they are sent to the web browser.

Junction is available as a Flatpak, so you can install it with `flatpak install flathub re.sonny.Junction`. When



Junction is especially useful on the web, because it can show the complete URL when you click a link, and can also flag insecure links.

you first start *Junction*, it offers to set itself as the default application for the web, which is a pretty good starting point.

To make *Junction* part of your daily workflow, just select the file in question, go to Properties, and change the default app for that file to *Junction*, and that's about it. The next time you open the file, *Junction* pops up and you can choose between various other apps that might be relevant from its menu.

Junction offers keyboard shortcuts for its commonly used functions. It requires virtually no configuration, though it does give you the ability to customise the list of apps it displays in the launcher.

STREAMING AUDIO PLAYER

Myuzi

Version: 1.13.3 Web: <https://gitlab.com/zehkira/myuzi>

Myuzi is an app for streaming music from YouTube. The best thing about it is that it has no ads and doesn't require an account.

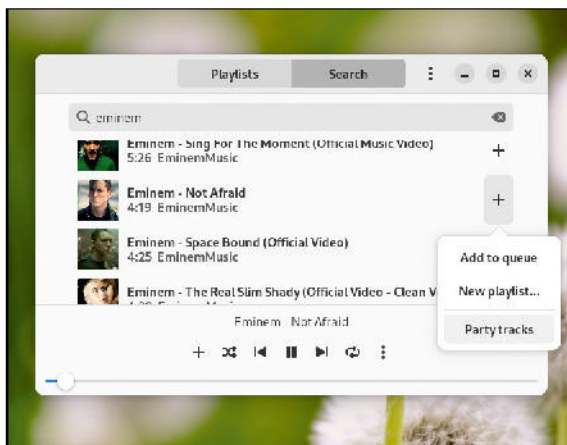
The app is available as a Flatpak, so you can grab it easily by using `flatpak install flathub com.gitlab.zehkira.Myuzi`.

As you'd expect, the app has a straightforward interface, with a couple of tabs at the top.

Since this is the first time you've launched the app and don't have any playlists yet, switch to the Search tab to find and queue some music.

Now enter some terms in the search box to find some music, such as the name of an artist, or a track's title. Once you've found something you'd like to listen to, simply click on the track and allow *Myuzi* to stream it for you.

The app offers all the conveniences that you'd expect from a music player. So, for instance, you can click the + icon adjacent to the name of a track to either add it to a new or an existing playlist, or just to add it to the playback queue. Similarly, while a track is



Toggle the Autoplay checkbox to get *Myuzi* to play tracks similar to the currently playing track.

being streamed, you can use the typical music playback controls from the bottom of the interface to play or pause the track, or jump forwards and backwards through the queue. You also have options to either loop a particular track, or ask *Myuzi* to shuffle the queue.

One odd design decision is the placement of the volume control, which is bundled inside the three-dot menu, along with the options to remove the current title from the queue, or to copy the complete URL to the playing song in the clipboard.

Once you've added tracks to a playlist, *Myuzi* gives you the option to download them all to local storage. This enables you to listen to the tracks on the playlists even when you're disconnected from the internet.

TRAIN SIMULATOR

Libre TrainSim

Version: 0.8**Web:** www.libretrainsim.org

Who doesn't love trains? With *Libre TrainSim*, you can experience being at the helm of one of these brilliant people movers. It puts you in the driver's seat of a virtual train with a first-person view of the controls and tracks. Passengers scurry across the platform to board your train, (*going on strike?—ED*) there are tracks from all across Germany, and you even get some shaking in the cabin in the latest version. Download the compressed archive for Linux from the project's website.

```
$ unzip Libre_TrainSim_Linux64_v0-8.zip
$ cd Libre_TrainSim_Linux64_v0-8
$ ./Libre_TrainSim.x86_64
```

You can load up a track straightaway and start controlling a train, but that isn't a good idea. Instead, we suggest you take advantage of the game's tutorials to learn the ropes, which includes everything from boarding passengers, closing the doors, starting the



engines and following the signals along the route to arriving at the stations on time.

If you want a challenge, you can disable easy mode for advanced training. Going through the tutorials and getting to grips with everything an engineer must do to make stops on time takes some fun out of the game, which we thought would just let us blow the horn as we raced along the tracks.

But once you get the hang of it, you don't just appreciate the effort it takes to get a train running on time, but also enjoy the leisurely pace. It's a nice step away from all the fast-paced titles and might be just what you need to bond with a four-year-old.

You want to create a new track or train? There is a wiki and YouTube playlist that teach you how to start.

If you just want to take in the experience of being in the driver's cabin, press **Ctrl+A** to turn on the autopilot.

POINT AND SHOOT

Infinite Shooter

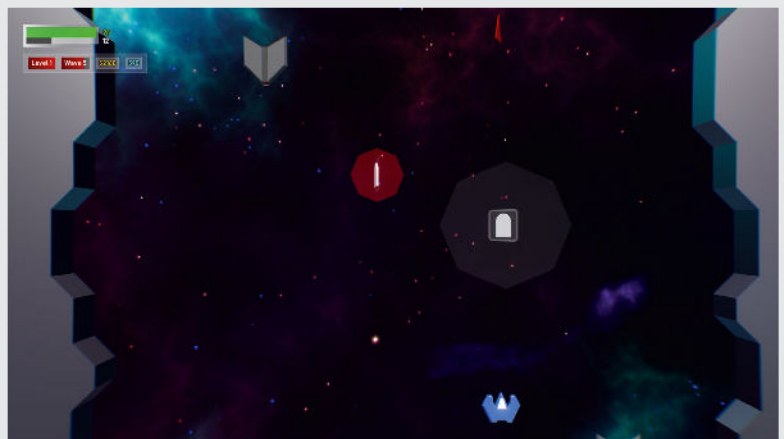
Version: 0.4.3 **Web:** <https://pastthepixels.itch.io/infiniteshooter>

If you think driving a train is cumbersome, piloting a plane should be infinitely more difficult, right? Not if it's a plane that fires lasers in *Infinite Shooter*. All you need to pilot this craft is control over the arrow keys to move your plane right and left to dodge enemy artillery fire, and to use the spacebar to fire your weapon.

As a 2D scroller, *Infinity Shooter* isn't the first to employ this type of gameplay. In fact, this genre has been done to death. But the mix of the right soundtrack, the flowing motion and flips of the aircraft, the futuristic-looking, geometrically perfect enemy spaceships and the simple-but-effective graphics make the gameplay surprisingly addictive.

You can install the game's Flatpak with **flatpak install io.github.pastthepixels.InfiniteShooter**.

The aim is to dodge enemy fire, destroy as many aircraft as you can, collect health and weapons when



you do so, and survive as long as possible. After every level, you get the option to trade in your spoils of war for upgrades, which don't make any sense to us. But if you know what to do with *The Slightly Legal Spoon*, *The Innocently Expensive House*, *The Neatly Important Hamburger* and such, this game is exactly what you need.

Thankfully, it's just as seductive to anyone who doesn't understand the upgrades. Just collect what you want, get back to the action, and survive for as long as you can.

As we said earlier, games like *Infinite Shooter* are a dime a dozen, but there's something refreshingly different about this that kept us hooked for a surprisingly long time.

The screenshot doesn't look it, but *Infinity Shooter* is a wonderfully entertaining point-and-shooter that'll keep you hooked.

OTP MANAGER

OTPClient

Version: 3.0.0 Web: <https://github.com/paolostivanin/OTPClient>

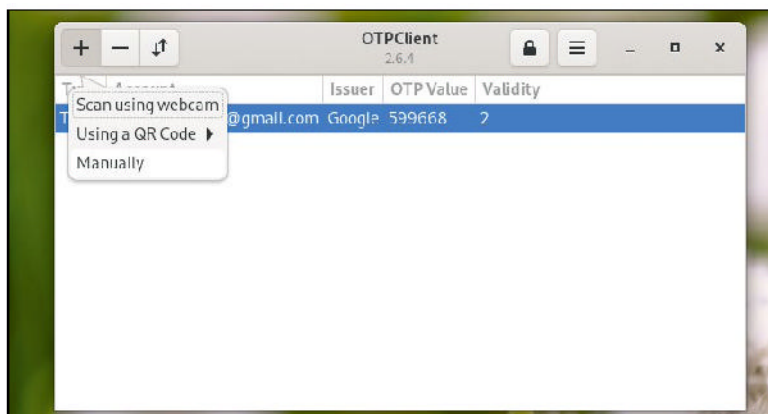
You're playing with fire if you aren't using two-factor authentication for any of your online accounts that support it. While it isn't exactly foolproof, having online apps verify your identity with an OTP token along with your password is still a lot better than relying on a password alone.

OTPClient is a nifty little app that helps you manage OTP tokens for various accounts and services, from within an intuitive app that has built-in encryption.

The app is packaged for several distributions, but you can just go ahead and grab the distro-agnostic Flatpak with `flatpak install flathub com.github.paolostivanin.OTPClient`.

On first launch, you have to choose an encryption password to secure the database for the app. Remember, there's no way to recover a forgotten password, so please be careful. After the first run, every time you start the app you are asked to enter this password to authenticate yourself.

Since this is the first time you've fired up the app, *OTPClient* asks whether you'd like to import some



If you are stepping away from the computer, either make sure you close the app or use the lock icon to prevent unauthorised use.

existing tokens, which it can from *Authenticator Plus*, or add new ones.

A typical workflow involves heading to the app that performs the two-factor authentication, and asking it to generate a QR code for the OTP login. Copy it to the clipboard or save it on your computer. As soon as you point *OTPClient* to the PNG image of the QR code, it automatically equips itself to generate OTPs to help you log into the app.

To show a token when you need to log into a web service or app, fire up *OTPClient* and click the desired row for the appropriate service. This displays the token and copies it to the clipboard, from where you can pass it to the app and authenticate yourself.

MULTI-MONITOR WALLPAPERS

HydraPaper

Version: 3.3.1

Web: <https://hydrapaper.gabmus.org>

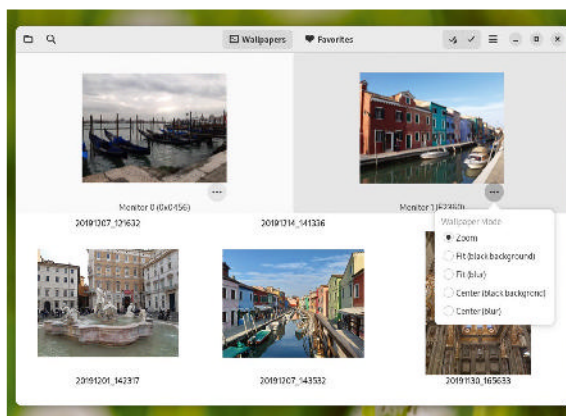
HydraPaper is a wallpaper manager designed to work around the lack of functionality on many desktop environments to set a different wallpaper for each monitor in a multi-monitor setup.

The app accomplishes this by scaling and merging different wallpapers into one and setting it as spanned across the monitors. *HydraPaper* currently supports all the popular desktop environments, including Gnome, Mate, Cinnamon and Budgie, and has experimental support for the Sway window manager as well.

The app is available for various distros, but again your best bet is to use the distro-agnostic Flatpak with `flatpak install flathub org.gabmus.hydrapaper`.

When you launch it, the app automatically displays all the monitors in your setup. Underneath this list, the app displays the contents of the **Pictures** folder by default. The folder icon in the top-left corner enables you to customise the location of the images you want to use as your wallpaper.

To set a wallpaper, first select the monitor from the list at the top. Then scroll through the list of images at



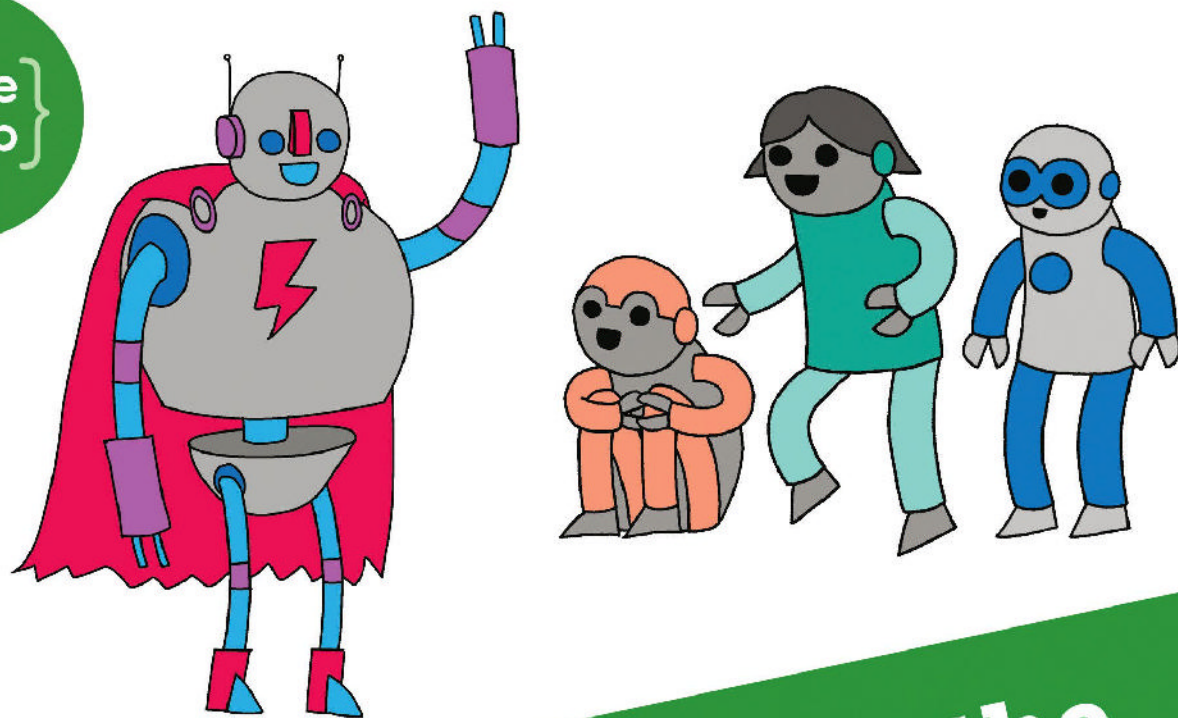
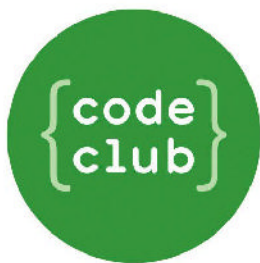
HydraPaper doesn't have very many configurable parameters, but can be handled completely from the CLI.

the bottom, and select the one you want to use as the wallpaper.

By default, *HydraPaper* zooms the image to fit the monitor. However, you can use the three-dot menu next to the monitor to set the wallpaper as centred or fitted, with either a black background or a blurred version of the selected wallpaper.

Now select the other monitor and repeat the process to set a wallpaper on that one. Once done, press Apply to activate the wallpapers. There's also an option that sets the selected wallpapers if your desktop environment has a dark mode.

One of the app's best options is to point it to your folder of wallpapers and select the Random option, which cycles through the images and changes the wallpapers across the monitors in your setup. **LXF**



**Can you help inspire the
next generation of coders?**



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

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

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

There are loads of ways to get involved!
So to find out more, join us at **www.codeclub.org.uk**

eBPF

Credit: <https://ebpf.io>

Part Six!
Don't miss
next issue,
subscribe on
page 16!

How to trace code directly with eBPF

Mihalis Tsoukalos explains how to take advantage of eBPF with a range of different programming languages.



**OUR
EXPERT**

Mihalis Tsoukalos is the author of *Go Systems Programming and Mastering Go*, and is currently working with *Time Series*.

T racing programming languages is a relatively difficult task, but we're not going to let that stop us, and are going to learn the principles of tracing C, Java and Bash shell scripts. First, though, let's talk a little more about eBPF internals and how to expose eBPF metrics to *Prometheus*.

You are not going to need this information on a daily basis, especially if you are not developing eBPF tools from scratch, but it might come handy when you want to know what is happening behind the scenes when executing eBPF code.

Compared to the original BPF, eBPF uses 64-bit words instead of 32-bit ones, has 11 registers instead of just two, offers more storage and can operate not only with packets but with multiple event sources. As mentioned in a previous eBPF tutorials, eBPF supports kprobes, which offer kernel dynamic instrumentation, uprobes for user-level dynamic instrumentation, and tracepoints that support kernel static instrumentation and are hard-coded into the Linux kernel code (`/usr/src/linux/include/trace/events/block.h`). Because tracepoints make your code more stable than kprobes, use tracepoints whenever possible. eBPF code attaches to one or more probes or tracepoints and executes the predefined bytecode each time a relevant event takes place. Put simply, kernel events are mapped to the BPF bytecode. Before mapping any BPF bytecode to kernel events, the BPF verifier, which is implemented using around 10,000 lines of C code, makes sure that the code is safe to execute. All this takes place in user space.

As with most programming languages, each eBPF program – which is in plain text format – is processed by a parser before constructing the abstract syntax tree, used for checking the validity of the code before it gets converted into bytecode and executed as machine code on the CPU. If you have misspelled a variable or a probe name, the semantic analyser is going to catch the relevant errors and print the appropriate error messages.

If the semantic analyser finds that everything is OK with the program, the code progresses into the code-generation phase, which has two main parts. In the first

```
32 def main():
33     b = BPF(text=BPF_PROGRAM)
34     my_event = b.get_syscall_fnname("getuid")
35     b.attach_kprobe(event=my_event, fn_name="handler_function")
36
37     # The port number for Prometheus to connect and get the metrics
38     start_http_server(2345)
39
40     # For Prometheus
41     nUsers = Gauge('number_of_users', 'Number of Users')
42     nTimes = Gauge('number_of_times', 'Number of Times')
43
44     while True:
45         nUsers_tmp = 0
46         nTimes_tmp = 0
47         ...
```

Figure 1: The main() function of the Python script used for getting eBPF metrics and exposing them to Prometheus via an HTTP interface.

part, the code is converted into LLVM IR (Intermediate Representation) and then it gets compiled into BPF bytecode using LLVM and Clang. The good thing here and the reason for creating the LLVM IR representation of the program is that LLVM has a BPF back-end (target) that converts LLVM IR into optimised BPF bytecode. The BPF code is sent to the kernel using the `bpf()` system call. *Bpftool* can help you look into the bytecode, the assembly and the disassembly code of running eBPF programs. Bear in mind that eBPF supports maps that are used for storing custom data. Due to their flexibility, maps are used as a key/value store as well as for storing eBPF histogram data.

Exporting eBPF metrics

Although getting metrics directly from eBPF is good, being able to store the metrics in a time series database opens the door to new possibilities, including the ability to go back in time and visualise data using software such as *Grafana*. In this section, we are going to learn how to expose eBPF metrics to *Prometheus* with the help of Python. The name of the Python script is going to be `export_prometheus.py` and it will contain code for both getting eBPF metrics and exporting them to *Prometheus*.

You should install the `prometheus_client` Python package using the `sudo -H pip3 install prometheus`

QUICK TIP

Tools such as `nm`, `objdump` and `readelf` can help you check whether the symbol table is present in a binary executable. If there is no symbol table, most of the time the best solution is to recompile the software with the appropriate command-line options.

`client` command. Otherwise, the script is not going to be able to find `prometheus_client`, the main reason being because we execute `export_prometheus.py` using root privileges.

Prometheus is going to be executed as a *Docker* image using a `docker-compose.yml` file, which you can find in the source code files for this tutorial in the *Linux Format* archive (<https://linuxformat.com/archives>). You can install *Prometheus* on your local machine if you are not very comfortable with *Docker*. What is important is the *Prometheus* configuration file (`prometheus.yml`). The part that reads the data from the Python script is:

```
- job_name: 'bpf'
  scrape_interval: 60s
  static_configs:
    - targets: ['host.docker.internal:2345']
```

The use of `host.docker.internal` as the target hostname is really important because it allows you to communicate with the Python script that runs on the actual Linux machine and not on a *Docker* container. Apart from that, do not forget to use the same port number value on both `prometheus.yml` and `export_prometheus.py`.

The important part of the Python code is the `start_http_server(2345)` statement that starts the HTTP server and the definition of `nUsers` and `nTimes` gauges for keeping the values of the metrics and exposing them. As a rule of thumb, when developing similar scripts, begin with the eBPF part, make sure that you get the metrics you want and then add the code for exposing them to *Prometheus*.

The Python code of the `main()` function of `export_prometheus.py` can be seen in Figure 1. The remaining code is about getting the desired eBPF metrics.

Running the Python script does not generate any output. However, connecting to the HTTP server that exposes the *Prometheus* metrics using *Curl* produces the next kind of output, which verifies that `export_prometheus.py` works as expected and exposes the desired metrics:

```
# HELP number_of_users Number of Users
# TYPE number_of_users gauge
number_of_users 2.0
# HELP number_of_times Number of Times
# TYPE number_of_times gauge
number_of_times 69.0
```

After that, we need to execute the *Prometheus Docker* image in order for *Prometheus* to begin collecting metrics. Going to the *Prometheus* web interface enables you to see the metrics in a simple plot, which is presented in Figure 2. Should you wish to create more professional plots, you should give *Grafana* a try – *Grafana* can read metrics from *Prometheus* and create truly professional output.

The good thing is that all Python scripts found in `/usr/share/bcc` can be easily modified to expose the collected metrics to *Prometheus* or another database.

The next section discusses the various types of programming languages in order to understand how eBPF deals with each type.

About tracing

Let's now discuss the general ideas and principles behind tracing programming languages with BPF. First of all, programming languages can be divided into



Figure 2: The Prometheus interface showing a simple plot of the two eBPF metrics exposed from the Python script.

QUICK TIP

Strip is a command-line utility that removes symbols from binary executable files and therefore makes these files smaller in size, which can be really important for embedded software or when there exist other kinds of limitations. However, Strip is an enemy of eBPF because it cannot operate on stripped binary files.

three main categories:

1. Compiled programming languages are the ones where the compiler generates a binary executable. This category includes programming languages such as C, C++, Rust and Go. As far as eBPF is concerned, this is the easiest programming language type to work with. However, as you can see in the boxout below, Go presents some difficulties, mainly because of its own concurrency model that uses goroutines instead of OS threads.
2. JIT (just in time) compiled programming languages are those where the compiler compiles source files into bytecode – just like eBPF does with its own code – that is compiled into actual machine code at runtime. This category includes programming languages such as Java, Erlang, Elixir and JavaScript.
3. Interpreted programming languages don't have a compiler – they use an interpreter instead. This

» TRACING GO CODE

Go is a very popular systems programming language that offers its own tracing tools – if the tracing tools that are provided by Go can do your job, use them. This includes tools such as the Go execution tracer and the Go deadlock detector.

We can use `sudo bpftrace -l 'uprobe:hw:*` to list the available uprobes of a Go executable file, which in this case is called `hw`. Remember that Go creates statically linked executable files that are not stripped by default. Using the generated output, we can identify the uprobes of interest and use them – if you don't know how to do that, revisit the eBPF tutorials in the previous issues of *Linux Format*.

However, although we can use eBPF to trace Go code, there are limitations on what we can do with eBPF and a Go binary executable, mainly because Go uses goroutines instead of OS threads as well as dynamic stack management, and has different function calling conventions. There is even the danger of eBPF crashing the target Go executable due to Go dynamic stack management. So, use eBPF cautiously with Go files. Going into more depth about Go and eBPF is beyond the scope of this tutorial, so just bear in mind that it might be easier to use the tracing tools provided by the Go ecosystem.



QUICK TIP

BPF Type Format (BTF) is a metadata format used for encoding debugging information for BPF programs and BPF maps. The output of `bpftool btf dump id <BTF ID>` shows the BTF-related information for a running eBPF program – this is pretty low-level information.

category includes programming languages such as Perl, Python and UNIX shell scripts.

Tracing C code

To start, we are going to learn about the tracing of C code, which is similar to what we have seen so far in this series of tutorials. However, what is truly important is to be able to identify whether a given binary file or a library that is being used contains the information needed for tracing. So, in this section, we are going to learn some techniques and utilities for finding out more information about binary files and whether they can be traced with eBPF or not.

WARNING: Inline C functions can't be traced, so do not put inline functions in your code if you want to be able to trace it. The other kind of C functions that can't be traced are those that are marked as unsafe to instrument. Stripped binary files can't be traced either.

So, the first thing to find out about the binary executable of a C project is whether it is stripped or not, which can be done with the help of the *File* utility. The section about the tracing of Bash scripts presents such an example. The `readelf` command can reveal differences in the information contained in the stripped and non-stripped versions of an executable file.

The output of the `readelf -s` command displays the symbol tables, using the Hello World! C program as an example in order to have a manageable output. Put simply, `strip()` removes the .symtab symbol table while leaving the .dynsym table intact.

In most Linux systems, there are versions of libraries containing extra debugging information that are helpful during development. On an Arch Linux machine, we can find these versions with the `sudo pacman -Ss debug` command – other Linux variables offer similar functionality. At this point, you might ask why you don't install the debug version of all libraries in the first place. The answer is simple: because the debug versions are bigger and run slower.

Another important detail that helps eBPF is the use of the frame pointer register for stack walking. This is enabled using the `-fno-omit-frame-pointer` parameter when compiling a C project.

Additionally, we can find out the USDT probes for static instrumentation of a program or library with the

use of *Bpfttrace*. This is illustrated in the next excerpt:

```
$ sudo bpfttrace -l 'uprobe:/lib64/ld-linux-x86-64.so.2:*'
uprobe:/lib64/ld-linux-x86-64.so.2: __nptl_change_
stack_perm
uprobe:/lib64/ld-linux-x86-64.so.2: __rtld_version_
placeholder
...
```

Other libraries produce a different output. For example, to get the list of supported uprobes from the `libc` library, you can execute `sudo bpfttrace -l 'uprobe:/lib64/libc.so.6:*'`, which at the time of writing generates about 2,577 lines of output.

You might ask, "What about C++?" As expected, tracing C++ code is almost the same as tracing C code, with some small differences. Firstly, symbol names are C++ signatures and, secondly, function arguments might behave a little differently. Additionally, C++ strings are objects instead of variables, so you might need to do some extra work when displaying the contents of a C++ string.

So, once again, bear in mind that if you are using a stripped binary executable, eBPF has no power over it. In that case, you should recompile your project.

Tracing Java code

The BCC package comes with tools that can trace Java code – the list of tools includes *Javacalls*, *Javaobjnew*, *Javaflow*, *Javastat*, *Javagc* and *Javathreads*. As part of JVM is written in C++, it can be instrumented as a compiled programming language. The list of available uprobes can be revealed with:

```
$ sudo bpfttrace -l 'uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so:*'
```

(The command depends on the path of the JVM library file, which depends on the Java version used.) Figure 3 shows a small part from the output of the command.

The output of *Javastat* looks similar to the following:

```
$ sudo /usr/share/bcc/tools/javastat -C
Tracing... Output every 1 secs. Hit Ctrl-C to end
PID CMDLINE METHOD/s GC/s OBJNEW/s
CLOAD/s EXC/s THR/s
5542 java -cp . -jar ./ta.o 0 0 0 0 0
22:09:19 loadavg: 1.62 0.86 0.39 3/619 6183
```

Unfortunately, discussing more about Java tracing is beyond the scope of this tutorial.

Bash scripts

In this section, we are going to use a Bash script called *traceMe.sh*. But first, let's begin by examining the Bash binary with the *File* tool:

```
$ file `which bash`
/bin/bash: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=6c75f9f0f273cf6549f078b042c0a3f5a04f0357, for GNU/Linux 4.4.0, stripped
```

It turns out that by default the Bash binary is stripped for performance reasons. This means we need to compile it on our own. On an Arch Linux machine, this requires the following commands:

```
$ cd /tmp
$ wget https://ftp.gnu.org/gnu/bash/bash-5.1.tar.gz
$ tar zxvf bash-5.1.tar.gz
$ cd bash-5.1
$ CFLAGS=-fno-omit-frame-pointer ./configure
$ make
```

Figure 3: Some of the uprobes that come with the `/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so` JVM library.

```
code sudo bpfttrace -l 'uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so:*' | wc
201      201    16037
code sudo bpfttrace -l 'uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so:*'
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: AsyncGetCallTrace
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JNI_CreateJavaVM
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JNI_GetCreatedJavaVMs
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JNI_GetDefaultJavaVMInitArgs
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ActiveProcessorCount
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AddModuleExports
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AddModuleExportsToAll
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AddModuleExportsToAllUnnamed
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AddReadModule
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AreNativesLoaded
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ArrayCopy
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_AssertionStatusDirectives
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_BeforeHalt
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_CallStackWalk
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_Clone
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetClassAt
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetClassAllIfLoaded
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetClassRefIndexAt
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetDoubleAt
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetFieldAt
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetFieldAtTfLoaded
uprobe:/usr/lib/jvm/java-18-openjdk/lib/server/libjvm.so: JVM_ConstantPoolGetFloatAt
```


» HOW TO APPROACH PROGRAMMING LANGUAGE TRACING

Here are some logical steps to follow that can help you with the analysis of a programming language:

1. Determine the kind of programming language that interests you: is it compiled, JIT or interpreted? Use the utilities, techniques and eBPF code presented in this tutorial accordingly.

2. Write a simple program in the desired programming language and check whether you can instrument it or not.

3. If you can't, search the internet for tools that can help you instrument that particular programming language. Do not use them immediately – just do your research for a better

understanding of whether or not there is sufficient eBPF support for the required programming language.

4. Determine whether the programming language has support for user-level statically defined tracing that comes with the installed packages and binaries. If not, check whether you need to

install other packages or recompile the binaries on your own.

5. Once again, write a sample program to verify that everything is working as you would expect.

6. Use uprobes for user-level software and kprobes for kernel-level software.

```
$ file bash
```

```
bash: ELF 64-bit LSB pie executable, x86-64, version 1
(SYSV), dynamically linked, interpreter /lib64/
ld-linux-x86-64.so.2, BuildID[sha1]=0542808441f9426b7
0bf73e75dad01213e69b2c4, for GNU/Linux 4.4.0, with
debug_info, not stripped
```

The first command is not necessary but it makes sure that you are not going to accidentally fill any important partitions on your Linux system. Just bear in mind that **/tmp** gets cleaned up after each system reboot and that we only need the new Bash binary – we are not even going to delete the installed version of Bash. The second command is for downloading the source code for Bash. The output at the end shows that we have done a great job and that the generated Bash binary is not stripped. A good place to put the new binary is **/usr/local/bin**. So, at this point, execute the `sudo mv bash /usr/local/bin` command in order to put it in **/usr/local/bin**. You now need to use the **/usr/local/bin/bash** path in your Bash scripts. As you are going to see in the next section, this path is also going to be used during tracing. After that, we are ready to begin tracing Bash scripts.

However, at this moment, it would be good to understand what is happening inside **traceMe.sh**. The **LXF** function, which contains two calls to **echo**, which is a built-in function, is executed four times due to the **for** loop. Therefore, **echo** is executed eight times in total. Additionally, there is a call to **sleep** at the end of the script.

The results

In this section, we are going to trace the **traceMe.sh** shell script. The simplest form of tracing is about counting the functions that were executed, which can be done with the **Funccount** tool:

```
$ sudo /usr/share/bcc/tools/funccount 'p:/usr/local/
bin/bash:*func*'
```

After running the previous command, you should execute **traceMe.sh** and stop **Funccount** when **traceMe.sh** has finished. Figure 4 shows the output from the tracing of **traceMe.sh**. What we see is that **execute_builtin_or_function** was called 12 times, which is the sum of the execution times of both **LXF** and **echo**. Additionally, **execute_function** was called

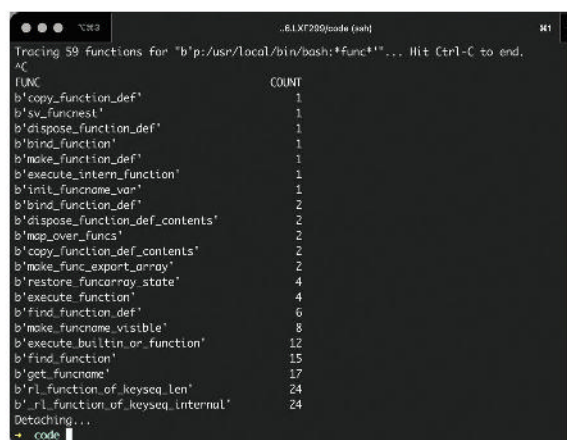


Figure 4: The output of the Funccount utility when tracing all ***func*** probes of the Bash shell executable.

four times due to the execution of **LXF** with the help of the **for** loop in the Bash script.

After that, we can use **Funccount** to trace all calls to **execute_* functions** (part of the generated output is omitted for brevity):

```
$ sudo /usr/share/bcc/tools/funccount 'p:/usr/local/
bin/bash:execute_*'
```

```
Tracing 31 functions for "b'p:/usr/local/bin/
bash:execute_*'"... Hit Ctrl-C to end.
```

FUNC	COUNT
b'execute_for_command'	1
b'execute_intern_function'	1
b'execute_function'	4
b'execute_builtin'	8
b'execute_simple_command'	13

Detaching...

The previous output shows that the count of **execute_builtin** is equal to the number of times **echo** function was executed. The count for **execute_simple_command** is equal to the count of **execute_builtin** plus **execute_function** plus one, which is added due to the use of the **sleep** command at the end of the script.

Finally, `sudo bpftrace -l 'uprobe:/usr/local/bin/bash:*'` returns all supported uprobes, which you can compare with the list of uprobes returned by `sudo bpftrace -l 'uprobe:/bin/bash:*'`, which is the Bash binary that comes with the Linux system.

As a last point, remember that not all programming languages or binary executable files can be traced. **LXF**

PYTHON

Recreating the classic Bomberman

When we said “go out with a bang”, **Andrew Smith** took it to heart and decided to blow more than just the blinking doors off!



**OUR
EXPERT**

Andrew Smith works as a developer for NHS Digital when he's not updating ancient gaming code.

QUICK TIP

Grab the code for this tutorial with:

```
git clone
https://
github.com/
asmith1979/
lxf299_
bomberman/
```

Bomberman, developed by Hudson Soft in 1983 and currently owned by Kanomi, is a game in which a player has to eliminate every other player on the game grid to win. We're looking at a clone of *Bomberman* created by Michal Sliva and Tomas Novak, the source code of which has been adapted for the benefit of this tutorial. The original source code for the project can be retrieved from <https://github.com/Forestf90/Bomberman>.

On a clean dev system, we need to install and set up Python 3.10. If you already have Python and PyGame installed, Python 3.8+ should be a good enough version to use with this tutorial. Type the following to install Python 3.10 and PyGame:

```
$ sudo apt-get update
$ sudo apt-get install python3.10
$ sudo apt-get install python3-pip
$ python3.10 -m pip install pygame
```

To make sure you are calling the correct *Pip* version, check in the **Scripts** folder of the Python installation folder. There may be a version number of *Pip* to use – *Pip* 3.8, for example.

Check the Python version and PyGame version, then *Git* clone from the repository:

```
$ git clone https://github.com/asmith1979/lxf299_
bomberman/
```

The source code and project can also be retrieved from the *Linux Format* archive. This tutorial focuses on

```
[andrew@localhost ~]$ git clone https://github.com/asmith1979/lxf299_
Cloning into 'lxf299_bomberman'...
remote: Enumerating objects: 94, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 94 (delta 1), reused 0 (delta 0), pack-reused 88
Receiving objects: 100% (94/94), 256.82 KiB | 1.30 MiB/s, done.
Resolving deltas: 100% (53/53), done.
[andrew@localhost ~]$ ls
Desktop Documents Downloads lxf299_bomberman Music OracleTest
[andrew@localhost ~]$ cd lxf299_bomberman/
[andrew@localhost lxf299_bomberman]$ ls
algorithm.py bomb.py enemy.py explosion.py game_lxf299.py game
[andrew@localhost lxf299_bomberman]$
```

A successful clone from the project repository. Once cloned, all the files are present, ready to view or edit.

the source code located in the folder called **lxf299_bomberman**, so if not already in that folder, type `$ cd lxf299_bomberman` to get into the folder and gain access to the Python source code.

Game project

In the root of the **lxf299_bomberman** folder, there exist some of the Python files we will be looking at in more detail as part of this tutorial, including **menu.py**, **game.py**, **enemy.py** and **player.py**. Also in the folder is a folder called **images**, which stores all the images that are used in the game. The files are:

- **menu.py** – contains all code for the mouse-driven menu system.

» DEPTH-FIRST SEARCH

Depth-first search (DFS) is an algorithm for searching or traversing through graph and tree data structures. In the context of this game project, the algorithm is used to help an enemy character choose what action to take depending on the current situation. On each level of the map, various actions are attempted, such as:

- **Place a bomb + move right**
- **Place a bomb + move left**
- **Place a bomb + move up**
- **Place a bomb + move down**

Sometimes these moves may not be possible because of a box crate, bomb or explosion nearby, so another move by the enemy character may be taken instead. The enemy character is placed in a corner of the game grid,

allowing analysis of the row of the grid they are situated in. The code for the depth-first search algorithm is located in the **Enemy** class implemented in **enemy.py**, mainly in the methods **dfs** and **dfs_rec**. The move that the DFS enemy character chooses to undertake is mainly based on the content outlay of the map – for

example, where the box crates, bombs, explosions and other enemy characters are placed on the map. In the Python script file **menu.py**, there is a boolean variable called **show_path** that by default is set to **True**. While this is true, the path of the enemy character is shown that the enemy character will travel along.

- **game.py** – contains the gameplay code
- **game_lxf299.py** – contains solution for human play win end condition
- **enemy.py** – contains all enemy character game code
- **player.py** – contains human player game code

Get Bombermanning

If you are currently in the **lxf299_bomberman** folder, type the following to run the game project:

```
$ python menu.py
```

This should display a basic mouse-driven menu system in which default settings have been applied so the game can be played straight away. Click on the Start menu option to begin the game. Use the arrow keys to move the game character around the game grid where needed and use Spacebar to plant a bomb. If you are not interested in playing the game until the end, press Ctrl+C to interrupt the program.

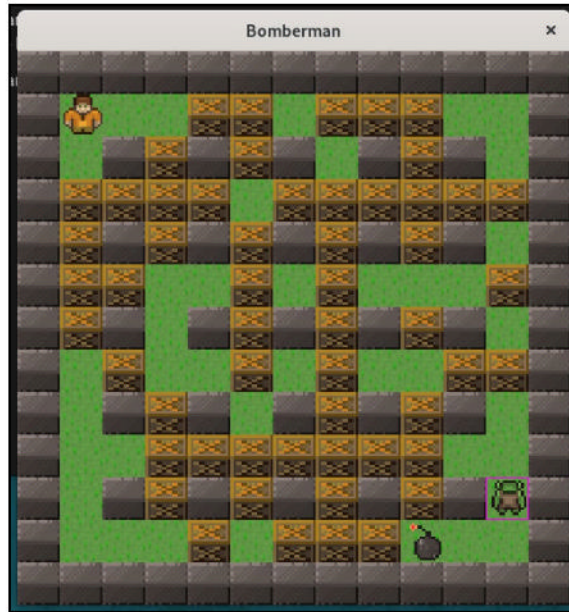
Worth mentioning at this point, if you do defeat the enemy player, there is currently no end condition to end the game, so you have to press Ctrl+C to stop the program. We will make a change to this later on. By default, the enemy game character is using the Dijkstra algorithm instead of using another algorithm called depth-first search (DFS – see the box bottom-left), which can be changed via the Options menu.

Game menu

Implemented as part of this tutorial is a mouse-driven game menu system, which is implemented in **menu.py**. Before we continue, open up **menu.py** in your chosen editor. It is the menu system that in turn starts the game program with settings selected from the menu. In **menu.py**, there are two menu systems implemented. The first one implemented is the menu we see first (see screenshot over the page), which lists the Start, Options and Exit choices. Another menu system that has been implemented is the Options menu, which mainly consists of defining whether each player of the game is human, a computer opponent with an AI algorithm or none (no player). Let's look at some of the code for the main game menu itself:

```
mainMenuEnd = False
while mainMenuEnd == False:
    # fill the surface white
    display_surface.fill(white)
    # copying the text surface object
    display_surface.blit(startGameText,
startGameTextRect)
    display_surface.blit(optionGameText,
optionGameTextRect)
    display_surface.blit(exitGameText,
exitGameTextRect)
    # iterate over the list of objects
    for event in pygame.event.get():
        ....
```

The variable **mainMenuEnd** is used as a boolean flag to identify when the **while** loop will end, which is currently set to the value of **False**. A **while** loop is used until the end user selects to either start the game or exit the game program, hence the line, **while mainMenuEnd == False:**. Below this declaration is the code to display the menu text shown when the main game menu is running. This then follows into the control segment initiated by a **for** loop, as shown in the



The game when it is started – human player at top-left and enemy AI player at the bottom-right – and the objects they have to blow out of the way by planting bombs.

code. We will now look at the code that detects which option the player is selecting:

```
...
if event.type == pygame.MOUSEBUTTONDOWN and
event.button == 1:
    pos = pygame.mouse.get_pos()
    if pos[0] >= startGameTextRect.topleft[0] and pos[0]
<= startGameTextRect.topright[0] and pos[1] <=
startGameTextRect.bottomright[1] and pos[1] >=
startGameTextRect.topleft[1]:
        game.game_init(show_path, player_alg, en1_alg,
en2_alg, en3_alg, TILE_SIZE)
        display_surface = pygame.display.set_mode((X, Y))

    if pos[0] >= optionGameTextRect.topleft[0] and
pos[0] <= optionGameTextRect.topright[0] and pos[1] <=
optionGameTextRect.bottomright[1] and pos[1] >=
optionGameTextRect.topleft[1]:
        optionsMenuProcess()

    if pos[0] >= exitGameTextRect.topleft[0] and pos[0]
<= exitGameTextRect.topright[0] and pos[1] <=
exitGameTextRect.bottomright[1] and pos[1] >=
exitGameTextRect.topleft[1]:
        mainMenuEnd = True
...
```

Under the **for** loop that detects events, you will find the above presented code. It starts by detecting a left-button mouse click has happened. After this, it tries to identify whether the left mouse button has been clicked on any of the main menu options. To help with detection of this, each of the menu options has been created with a rectangle area around it, so for the option to start a game, the top-left corner of the menu option can be accessed by **startGameTextRect.topleft[0]** to check the value of the top-left corner of the Start Game menu option. The **pos[0]** and **pos[1]** represent the mouse pointer position, x and y respectively. So, given this, it is possible to detect whether the mouse pointer is within an area represented by each menu option.

The code for the main menu system has not been put in a function as it's the first code executed. As can

QUICK TIP

When writing a program that uses graphics resources such as this one, create a folder called "images" that can store all the images used for the game project so that it is organised in a structured way.



be seen from the previous code, if the first option is selected, the game program is started with various arguments passed in, which are the players in the game. The second menu option takes us to the Options menu, which is implemented in a function called `optionsMenuProcess()`. The third menu option exits from the game program.

As just previously mentioned, the Options menu code is located in the `optionsMenuProcess()` function. This is another **while** loop nested in another **while** loop (the main game menu **while** loop), with another loop end condition. Let us take a look at the code for the Options menu:

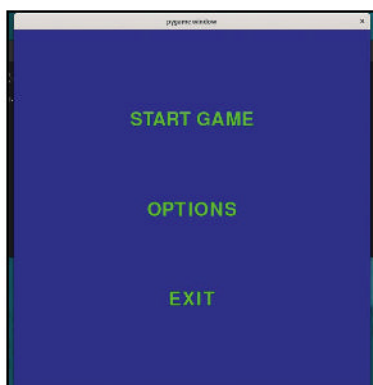
```
while optionsMenuEnd == False:
    display_surface.fill(blue)
    display_surface.blit(playerTwoText,
        playerTwoTextRect)
    display_surface.blit(playerThreeText,
        playerThreeTextRect)
    display_surface.blit(playerFourText,
        playerFourTextRect)
    display_surface.blit(backText, backTextRect)
    for event in pygame.event.get():
        ...
```

As can be seen from the above code, it follows the same structure and flow as the main game menu, where there is a **while** loop boolean variable to determine when the loop ends. In this case, the name of the variable is `optionsMenuEnd`, which is set to **False** in the beginning before the **while** loop starts. Also notice that the Options menu text is output at this stage as well, same as with the main game menu just before it enters into a **for** loop to detect any input events that have happened. Let's look at the control statements where the player selects an option from the Options menu.

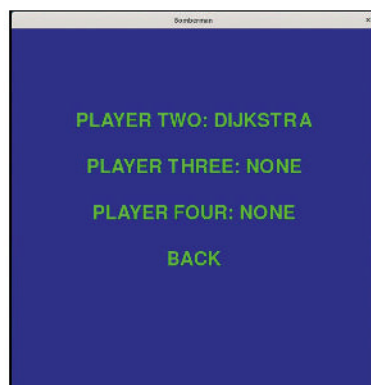
```
if event.type == pygame.MOUSEBUTTONDOWN and
    event.button == 1:

    pos = pygame.mouse.get_pos()

    if pos[0] >= playerTwoTextRect.topleft[0] and pos[0]
    <= playerTwoTextRect.topright[0] and pos[1] <=
    playerTwoTextRect.bottomright[1] and pos[1] >=
    playerTwoTextRect.topleft[1]:
        if en1_alg == Algorithm.DIJKSTRA:
            playerTwoValue = 'PLAYER TWO: DFS'
            en1_alg = Algorithm.DFS
```



The main game menu enables you to start the game, go into the Options menu or exit the game – all mouse-driven.



The Options menu can be used to change the AI algorithm of the enemy player or include more enemy AI players on the game grid.

```
elif en1_alg == Algorithm.DFS:
    playerTwoValue = 'PLAYER TWO: NONE'
    en1_alg = Algorithm.NONE
elif en1_alg == Algorithm.NONE:
    playerTwoValue = 'PLAYER TWO: DIJKSTRA'
    en1_alg = Algorithm.DIJKSTRA
```

```
playerTwoText = font.render(playerTwoValue,
    True, green, blue)
```

Only part of the event code for the Options menu has been given as it's quite lengthy and the same logic is applied in each case selection. From the above example code, the left mouse button click is first detected, very much like the main Options menu. Following on from this, a check is done to see if the end user is clicking in the range of the Player Two option in the Options menu. The logic here does now become a little more complicated than it did for the main Options menu. Here we have a system that effectively scrolls through different options that Player Two can be, which could be any one of the following:

- **AI Player: DFS**
- **AI Player: Dijkstra**
- **None (no player)**

After an option has been decided or selected, the Player Two text is updated to reflect this. Note that there is no player one option because player one is predefined as being human.

Gameplay code

The project was created to pitch at least two or more AI players (with different algorithms) against a human player, mainly to demonstrate the effectiveness of the depth-first search algorithm versus the Dijkstra algorithm. The main gameplay code is in `game.py`.

Open `game.py` and, reading down from the top of the script, there are various aspects set up, including the game grid, images used for animation in the game and any text used to display during gameplay. You will also see there is a function called `game_init`, which is called from `menu.py` after the selections have been made. This function is used to take in the players that will be present in the game and further set up the game characters in relation to the game grid and to further process the graphics resources used for the map layout, including objects such as bombs. Other relevant functions present in `game.py` include:

- **draw()** – renders the game graphics
- **generate_map()** – generates the grid layout
- **main()** – main gameplay loop
- **update_bombs()** – explodes bombs if time limit is reached
- **game_over()** – processes end game result

The main function of interest out of this list is `main()`, which contains the main game processing **while** loop. The main game processing **while** loop is set to only end if the player is not alive. See the following example code:

```
while player.life:
    dt = clock.tick(15)
    for en in enemy_list:
        en.make_move(grid, bombs, explosions, ene_
            blocks)
```

While the player is alive, the game program does not end. This may not be a problem if an enemy AI



Here the enemy AI has won the game using the Dijkstra algorithm. The human player has lost the game. The name of the algorithm that the enemy player was using is displayed.

player defeats the human player – then the program ends. However, if the human player defeats an AI player, then the program runs continually regardless. To solve this, we need to change the current way the main **while** loop ends in **main()** which brings us to the next part of this tutorial. A solution for this is located in **game_lxf299.py** if you wish to learn by viewing rather than by doing. We will go through the changes step by step.

Player wins

Go to the top of **game.py** and after the **import** and **from** statements, type the following:

```
OUTCOME_PLAYER_WIN = 1
OUTCOME_ENEMY_WIN = 2
game_outcome = 0
```

This code initially defines two end outcomes of the game. Either the human player wins or an enemy player wins. The outcome if either enemy AI player wins is already accounted for. Initially, the **game_outcome** is set to **0** because to start with nobody has won the game. Now scroll down to the **main()** function so we can write in an end condition for when the human player wins. Change the main **while** loop code to the following:

```
game_end = False
def main():
    global game_end
    global game_outcome
    global OUTCOME_ENEMY_WIN
    global OUTCOME_PLAYER_WIN
    generate_map()
    while game_end == False:
```

The **global** keyword identifies that we are including a global variable declared outside of the function. The important part here is to notice that we have changed the **while** loop condition to **game_end**, which is initially

set to **False**. Now scroll down to just after the following code and press the Enter key twice to provide some more space:

```
if movement:
    if player.frame == 2:
        player.frame = 0
    else:
        player.frame += 1
```

Now type the following:

```
# Add end condition
if player.life == False or len(enemy_list) == 0:
    if len(enemy_list) == 0:
        game_outcome=OUTCOME_PLAYER_WIN
    if player.life == False:
        game_outcome=OUTCOME_ENEMY_WIN
    game_end = True
```

Now, if you run the game program and defeat the enemy, the game program should end and take you straight back to the game menu. The solution presented as part of this tutorial is very simple and direct. It could be further improved by implementing further code in the **game_over()** function to identify a human player has won instead of just taking you straight back to the main game menu. Also, this solution does not account for more than one human player being in the game, so the solution needs to be further modified. Other improvements to this project could include:

- Adding sound effects
- Improved coding (replacing hard-coded values, etc)
- Adding another search algorithm for an enemy player
- Adding an escape routine to exit the game part or midway through – pressing **Escape**, for example

If you are new to coding, you may require more time to read through and understand the code better before fully completing the tutorial. 📖

» DIJKSTRA ALGORITHM

The Dijkstra algorithm is used to find the shortest path between two nodes in a graph. With regard to this game project, the algorithm is first used to find the shortest distance to a box crate to bomb and then the shortest distance to safety. If the player is nearer to the enemy character than a box crate is, then the shortest path is calculated between the enemy character and another player, which could be a human player or another AI player. Once all the box crates on the game map have been eliminated, the other enemy players on the game map (if they are still alive) is targeted with the shortest possible route to them. To see the route that the enemy Dijkstra character calculates, leave the variable named **show_path** set to **True** – this is set in the Python script file **menu.py**. As well as attacking enemy players, the algorithm is also used to attempt to escape from enemy defeat by trying to calculate the shortest distance to safety behind a boulder that can protect against a bomb blast. All the decisions are based on the current grid or map layout of bombs, players, explosions or boulders, which is updated in real time during play. The implementation of this algorithm can be found in the **Enemy** class located in **enemy.py** in the class methods **create_grid_dijkstra** and **dijkstra**.

» BLOW YOUR MIND EVERY ISSUE! Subscribe now at <http://bit.ly/LinuxFormat>

NEXT MONTH

TOP 100 TOOLKIT

We ring in our 300th issue of hot open source software coverage by celebrating the best apps and tools to use!



LXF300
will be on sale
Tuesday
7th March
2023

Mastodon social

What's all this fuss about Twitter? We're too busy over here with the cool kids playing in the Mastodon pools...

Best distro ever?

We pit the biggest-name distros head to head to discover which one is king! Yes, of course everyone is going to disagree.

Hurrah for Firefox!

A look back over the last 20 years as we give thanks that there's a dedicated open source browser we can all trust!

A Hurd kernel to use

We make more terrible jokes about the 'other' open source kernel from GNU and explain how you can try it out.

LINUX
FORMAT
The #1 open source mag

Future Publishing Limited,
Quay House, The Ambury, Bath, BA1 1UA
Email contact@linuxformat.com

EDITORIAL

Editor-in-chief Neil Mohr
Art editor Fraser McDermott
Production editor Katharine "The Punisher" Davies
Group editor-in-chief Graham Barlow
Group art director Jo Gulliver

Editorial contributors

Mike Bedford, Jonni "late as usual" Bidwell,
Neil Bothwick, Shane Downing, Nate Drake,
Will Freeman, Karsten Günther, Dave James,
Matthew Holder, Jon Masters, Nick Peers,
Aaron Peters, Les Pounder, Michael Reed,
Mayank Sharma, Shashank Sharma,
Andrew Smith, Mihalis Tsoukalos

Cover illustration

Magictorch.com
Android is a trademark of Google LLC. Fedora and the Infinity
design logo are trademarks of Red Hat, Inc.
Ubuntu is a trademark of Canonical Limited. We are not endorsed
by or affiliated with Canonical Limited or the Ubuntu project.
Raspberry Pi is a trademark of the Raspberry Pi Foundation.
Tux credit: Larry Ewing (lewing@isc.tamu.edu).

Content production

Adequate Media Limited

ADVERTISING

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

Head of print licensing Rachel Shaw
Linux Format is available for licensing and syndication.
To find our more contact us at licensing@futurenet.com
or view our content at www.futurecontenthub.com.

NEW SUBSCRIPTIONS & BACK ISSUES

Web www.magazinesdirect.com
UK 0330 333 1113 World +44 (0) 330 333 1113

EXISTING SUBSCRIPTIONS

Web www.mymagazine.co.uk
UK 0330 333 4333 World +44 (0) 330 333 4333
Subscription delays: Disruption remains within UK
and international delivery networks. Please allow up
to seven days before contacting us about a late
delivery to help@magazinesdirect.com

CIRCULATION

Head of newstrade Tim Mathers

PRODUCTION AND DISTRIBUTION

Head of production UK & US Mark Constance
Production project manager Clare Scott
Senior ad production manager Jo Crosby
Digital editions controller Jason Hudson

THE MANAGEMENT

Editorial director technology group Paul Newman
Content director Joe Cox
Head of art and design Rodney Dive
Design director Brett Lewis
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

Linux® is the registered trademark of Linus Torvalds in the US and other countries.
GNU/Linux is abbreviated to Linux throughout for brevity. Where applicable,
code printed in this magazine is licensed under the GNU GPL v2 or later. See
www.gnu.org/copyleft/gpl.html. All copyrights and trademarks are recognised
and respected.

Disclaimer All contents © 2023 Future Publishing Limited or published under
licence. All rights reserved. No part of this magazine may be used, stored,
transmitted or reproduced in any way without the prior written permission of the
publisher. Future Publishing Limited (company number 2008885) is registered in
England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All
information contained in this publication is for information only and is, as far as we
are aware, correct at the time of going to press. Future Publishing Limited cannot
accept any responsibility for errors or inaccuracies in such information. You are
advised to contact manufacturers and retailers directly with regard to the price of
products/services referred to in this publication. Apps and websites mentioned in
this publication are not under our control. We are not responsible for their contents
or any other changes or updates to them. This magazine is fully independent and
not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have
the necessary rights/permissions to supply the material and you automatically
grant Future Publishing Limited and its licensees a licence to publish your
submission in whole or in part in any/all issues and/or editions of publications, in any
format published worldwide and on associated websites, social media channels
and associated products. Any material you submit is sent at your own risk and,
although every care is taken, neither Future nor its employees, agents,
subcontractors or licensees shall be liable for loss or damage. We assume all
unsolicited material is for publication unless otherwise stated, and reserve the right
to edit, amend, adapt all submissions. All contents in this
magazine are used at your own risk. We accept no liability for any
loss of data or damage to your systems, peripherals or software
through the use of any guide.

We are committed to only using magazine paper derived from
responsibly managed, certified forestry and chlorine-free
manufacture. The paper in this magazine was sourced and
produced from sustainable managed forests, conforming to
strict environmental and socioeconomic standards.

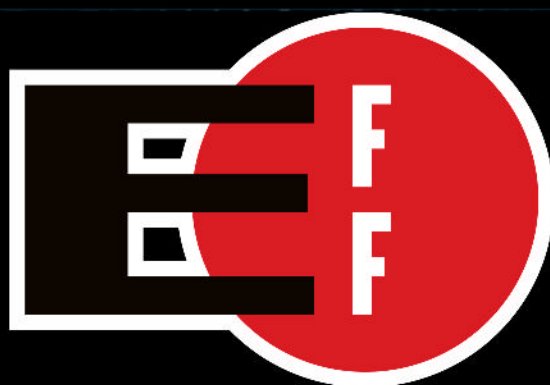


Future is an award-winning international media group and
leading digital business. We reach more than 57 million
international consumers a month and create world-class
content and advertising solutions for passionate
consumers online, on tablet & smartphone and in print.

FUTURE **Connections. Creators. Experience. Masters.**

Future plc is a public
company quoted on the
London Stock Exchange
Symbol: FUTURE
www.futureplc.com

Chief Executive: Zillah Byng-Thorne
Non-Executive Chairman: Richard Huntingford
Chief Financial and Strategy Officer: Penny Liddle-Stand
Toll +44 (0)2025 442 244



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



**THE
BRAIN
TUMOUR
CHARITY**

A CURE CAN'T WAIT

9021

BRAIN TUMOURS MOVE FAST. WITH YOUR HELP, WE CAN TOO!

We're working to create a future where brain tumours are curable.
We urgently need your help to accelerate research.

Text DEFEAT5 to 70507 to donate £5, please help us to find a cure.

thebraintumourcharity.org

© The Brain Tumour Charity 2020. Registered Charity in England and Wales
(1150054) and Scotland (SC045081)



Registered with
**FUNDRAISING
REGULATOR**