BUILD YOUR OWN SERVER

How to run a secure VPS » Use Apache » Let's Encrypt » Deploy Docker » Install services » Optimise databases

FREE DOWNLOAD ZORIN OS 16

LINUX FORMAT

The #1 open source mag

PLUS!

- FASTER WEBSITES
- RUN THE TANDY TRS-80
- ROLL YOUR OWN DISTRO

LINUX ♥ WINDOWS

Linux now comes built into Windows! Discover how easy it is to get started

67 pages of tutorials & features

Create classic photo effects with GIMP filters and tools

Achieve atomic-accurate time with the GPS system

Transfer files faster over your local network

WEB REQUESTS

Quickly code a REST server in Go

CODING GALAXIAN

Start writing a classic multiplayer space shooter

LXF November 2021

FUTURE

COVID

WHO IS MOST AT RISK? WHAT CAN PREVENT IT?

thomaswimberly.com @thomaswimberly @thomaswimberlystudios

PLEASE TAKE THE SURVEY

qmul.ac.uk/covidence

Queen Mary University of London

THE UNIVERSITY of EDINBURGH

Swansea University Prifysgol Abertawe

QUEEN'S UNIVERSITY BELFAST

SIGN UP

LINUX FORMAT

Corporate Patron Free Software Foundation

MEET THE TEAM

This issue we're trying to get Windows users to try out Linux. What cool FOSS thing do you think could tempt Windows users over to Linux?

Jonni Bidwell
If Reddit is anything to go by then all it takes is a screen grab of an i3 desktop with some sort of sci-fi/manga background, a translucent terminal and Conky proudly showing the handful of running processes consuming but a smidgeon of the machine's gargantuan memory.

Les Pounder
With Valve's upcoming Steam Deck running Arch Linux, we are going to see a big push for Linux gaming, thanks to Steam and Proton. These advancements will filter down to Linux in general, so you will soon be able to frag newbs via your Tux-powered rig.

David Rutland
If macOS is a Tesla and Windows a Honda Civic, Linux puts an entire dealership at your disposal with distros catering from new drivers to mechanics tinkering under the hood. Or build your own with jacked-up wheels and no windscreen and all without paying a thing.

Shashank Sharma
Abundant software choices, crazy-cool projects across different genres, excellent documentation, helpful user community are all good reasons to try a Linux distribution. If you've never ventured beyond Windows, just the possibility of something different should be tempting enough.

Alexander Tolstoy
I can quickly recall that lots of FOSS things simply run more smoothly and better on Linux than Windows – such as OpenCV-based and machine learning projects, modern Javascript frameworks. I believe that Linux is a first-class OS for most research, scientific and development needs.

Windows Insider

So after saying it would never exist, Microsoft felt it necessary to launch Windows 11. Why do we care? Well we don't, but it's neat to note that the Linux kernel now comes baked into Windows as part of WSL 2.0. It means people can try out and use Linux and its FOSS ecosystem more easily than ever before.

We know many see this as more Microsoft 'embrace and extinguish', but this is different. Microsoft can't own the Linux kernel, can't shut it down – all it can do is attempt to make access to Linux as easy as possible from within Windows. My view is this simply opens Linux to a wider audience, which has to be a good thing. Right?

In this issue Jonni – the poor boy – has been playing with Windows 11 to see if it will play nicely with Linux. He's looking at dual-booting it, the TPM 2.0 situation, Secure Boot issues and explores the latest Windows Subsystem for Linux 2.0 to show how you can run a native GUI-based Linux install right in the heart of Windows with no performance penalties – other than it running Windows, of course.

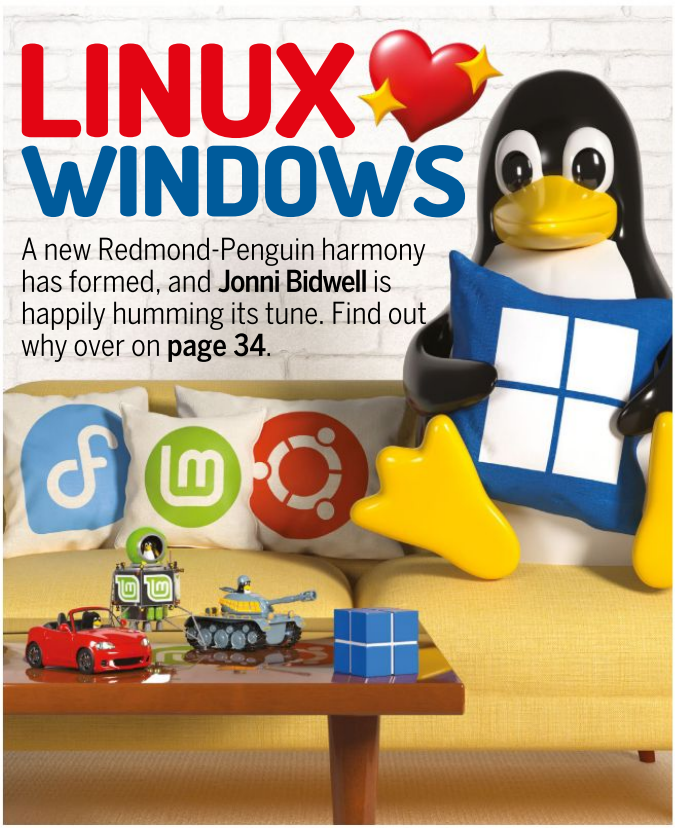
As we know most of you aren't running Windows, we have a pile of superb open source Linux fun in the rest of the magazine. We're kicking off a new series looking at setting up a server (home or hosted) and what you can run on it, *Roundup* static website generators, chat to the AlmaLinux developers on replacing CentOS, use GPS for atomically accurate time, explore *GIMP* photo filters, and loads more – so enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com

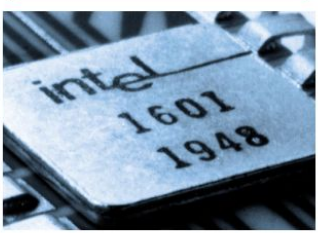
Contents

REVIEWS



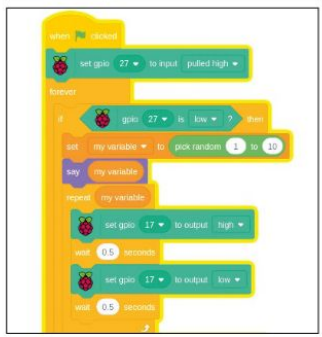
ROUNDUP

IN-DEPTH



CONTENTS

Pi USER



CODING ACADEMY

REGULARS AT A GLANCE

ON YOUR FREE DVD
• Zorin OS 16
• Lakka • LibreElec
See page 96

TUTORIALS

IN-DEPTH

Newsdesk

THIS ISSUE: Sir Clive Sinclair » Steam Deck improvements » WSL malware » EU RISC-V chips » Old Ubuntu gets extended support

OBITUARY

Remembering the legacy of Sir Clive Sinclair

Inventor and pioneer was behind the iconic ZX81 home computer, plus the Sinclair QL – which Linus Torvalds learned to program on.

Sir Clive Sinclair has died at the age of 81. After leaving school at 17, Sinclair worked for a while as a technical journalist before creating Sinclair Radionics in 1961. Its first product was the Sinclair Micro-amplifier, released in 1962. After almost a decade of creating audio equipment, the company began creating popular pocket calculators that were considerably smaller than the competition.

However, Sinclair was best known for effectively inventing home computers with the ZX80. Released in 1980 costing £79.95 as a kit, or £99.95 pre-assembled, it was drastically cheaper (and smaller) than any other computer at the time, and it meant having a computer for the home was finally affordable for many people. Selling 100,000 units the ZX80 was moderately successful, its follow-up the ZX81 was far more of a hit selling 1.5 million.

In 1982, Sinclair Research, the company that succeeded Sinclair Radionics, released the ZX Spectrum. Selling over 5 million units this range of computers ushered in a generation of developers who are still shaping the software we use and the games we play today.

One of those was Linus Torvalds, who created the Linux kernel, and without whom this fine magazine wouldn't exist. In an interview at LinuxCon 2010 (which you can watch at **https://bit.ly/3zHFIWu**), Torvalds explains how he learnt to code on a Sinclair QL, linking the late Sir Clive Sinclair to everyone's favourite open source operating system, Linux. While it would be rather silly to claim that without Sinclair and his home computers they'd be no such thing as Linux – as Torvalds would have almost certainly have used an alternative – the fact that Sinclair made home computing an affordable option for so many people has rightly been celebrated.

While not everything that Sinclair released was a success (his most infamous failure the Sinclair C5 electric car was too far ahead of its time), his impact on computing cannot be overstated. While he never fully succeeded in turning Cambridge into a UK-based Silicon Valley, for a while he was behind the best-selling computers in the world, and his death led to tributes from the likes of Elon Musk, who tweeted (**https://bit.ly/lxf282musktweet**) "RIP, Sir Sinclair. I loved that computer." Meanwhile, Satya Nadella, CEO of Microsoft, posted (**https://bit.ly/lxf282nadellatweet**) "RIP Sir Clive Sinclair. Your innovations democratized computing and inspired so many, including myself. I vividly remember my first computer, a ZX80, and the sense of wonder and empowerment I felt."

The ZX Spectrum was perhaps Sir Clive's most popular invention, and inspired many people to get into coding.

MICROSOFT CEO PAYS TRIBUTE

"I vividly remember my first computer, a ZX80, and the sense of wonder and empowerment I felt."

The less well know Sinclair QL (Quantum Leap) is the computer Torvalds learnt to program on.

GAMING

Linux games get faster

AMD has been working on making Ryzen processors work better with Linux.

Back in **LXF280** we reported on how Valve was preparing to launch its Steam Deck handheld console, which runs Linux in the guise of its Arch-based SteamOS 3.0 distro, and it appears that work is bringing benefits to other Linux gamers as well. As Tom's Hardware reports (**https://bit.ly/lxf282valveamd**), Valve has been working with AMD to create a new CPU driver that will improve the performance and power efficiency of Zen-based AMD processors on the Linux platform – which includes the Steam Deck.

As AMD developer Ray Huang explained at the recent X.Org Developers Conference (XDC2021), Valve discovered issues with the current ACPI CPUFreq driver, which was impacting performance with games running on Proton. Proton is a fork of *Wine* that was co-created by Valve to enable Windows-only games to run on Linux. Valve is understandably keen to make sure that everything runs as smoothly as possible, which meant creating a new driver to replace the ACPI driver that had originally been created by Intel for its first-generation Core processors.

So, according to Huang in a presentation (which you can view at **https://bit.ly/lxf282xdcpres**), Valve contacted AMD, and the companies worked together on a new CPPC driver that will use a specialist P-state driver to control CPU clock speeds depending on workload. As Tom's Hardware reports, early results are very promising, with a Ryzen 7 5750G system seeing a boost to performance per watt of 10-25 per cent, with games such as *Horizon Zero Dawn* seeing a particular boost. Best of all, this driver – which AMD is planning to get into the official Linux kernel once it's stable – will benefit all Linux gamers with AMD hardware, not just Steam Deck owners.

The forthcoming Steam Deck handheld console runs on Arch Linux, and it's bringing benefits to all Linux gamers.

SECURITY

WSL malware discovered

New threat could compromise Windows machines.

Windows Subsystem for Linux (WSL) is one of the best recent additions to the Windows OS as it adds everyone's favourite kernel to Windows. In this very issue we look at WSL 2.0, which is built into Windows and enables you to run Linux executables natively in Windows 10 and Windows 11, with WSL 2.0 including the Linux kernel.

It's handy for devs working on Linux projects but using a Windows PC, for example, and it's certainly been one of the more popular aspects of Microsoft's embrace of open source. However, security researcher Black Lotus Labs has discovered malware that uses WSL to avoid being detected by antivirus tools. In a blog post outlining their findings (read it at **https://bit.ly/lxf282wsl**), the researchers reveal they have "recently identified several malicious files that were written primarily in Python and compiled in the Linux binary format ELF (Executable and Linkable Format) for the Debian operating system."

These files included a payload for target PCs that was either embedded in the file, or downloaded from a remote server, and then injected into a running process via Windows API calls. While Black Lotus Labs said that this method "was not particularly sophisticated", because it was using the WSL environment – which up until now was rarely used in malware attacks – antivirus tools were having trouble detecting it. The researchers hope by publishing their findings it will make detecting threats using WSL easier in the future.

OPINION

THE MOST WANTED

Matt Yonkovit
is Percona's Head of Open Source Strategy and a member of SHA (Silly Hats Anonymous).

"PostgreSQL is currently ranked the "most wanted" database according to Stack Overflow, and DB-Engines crowned it DBMS of the year 2020. By the time you read this, PostgreSQL 14, the latest version of the open source database and is a great example of the power of open source. The licence allows it to be shared, used, forked and built upon. Hundreds of different companies, commercial and open source, have created spin-off versions.

In an effort to ensure long-term supportability and quality, maintainers have tightened control over what makes it officially into a release. While this may not sit well with all contributors, the great news is that PostgreSQL's architecture was designed to allow new additions and features to be built and shipped as extensions.

One reason open source is so great is that it brings together ideas from people with differing perspectives, adopting the best of them. Not everyone will be happy with every outcome, but as long as we continue to discuss, debate and evolve, we can all be successful.

With the launch of version 14, the momentum behind PostgreSQL shows no sign of abating."

OPINION

SINKING FEELING

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

"I was talking to a company recently who told me they'd written their own system monitoring application. I was curious: why would they do that?

No one outside the organisation knows about it, so new staff have to be trained on it. They can't Google any questions they have. Interfacing with other systems (for example, to handle out-of-hours alerts) requires more coding. And it isn't even that good.

Reading between the lines, I suspect the sunk cost fallacy is in play here. They've spent so much time and money developing it that throwing it away feels wasteful. So they spend even more time and money on it...

Open source software excels in some areas, and system management, including monitoring, is one such area. I would accept that sometimes commercial products look prettier, but the functionality is there in both open source and commercial offerings.

Some businesses have a philosophy of avoiding 'Not Invented Here' software. That's seldom justified these days (if indeed it ever was). If you're using home-grown software to manage your Linux systems, you are almost certainly wasting money."

PROCESSORS

European RISC-V chip gets closer

Test samples of the EPI's RISC-V processor have been delivered.

CREDIT: EPI

Close-ups of the test samples.

The European Processor Initiative (EPI) is a project made up of 28 partners from 10 EU countries, including the BMW Group, the Barcelona Supercomputing Centre (BSC), the University of Zagreb and more. It aims to allow the EU to gain independence in HPC chip technology, and recently announced (**https://bit.ly/lxf282epiannouncement**) that samples of its EPAC1.0 RISC-V Test Chip have now been delivered, and initial tests look to have been very promising. Based on the RISC-V open source architecture, the chip is made up of power-efficient and high-throughput accelerator cores named EPAC (European Processor Accelerators). The 22nm test chips were made at GlobalFoundries, which used to be part of AMD, and it means the EU joins a growing list of countries and governments that are turning to RISC-V to create their own processors.

The pace of the EPI's progress is certainly impressive; the EPAC design was only proven to work in March, and now a chip using the technology has been shown to boot Linux successfully. Make sure you check out the EPI Project's website for more information at **https://bit.ly/lxf282epi**. We'll be keeping a close eye on this one...

FILE SYSTEMS

Linux 5.15 NTFS

The forthcoming kernel will come with a driver for better Linux support of the NTFS filesystem.

Paragon Software has announced that its previously commercial-only NTFS3 kernel driver will now be mainlined, and should be coming with Linux 5.15. NTFS is a proprietary file-system designed by Microsoft for its Windows operating systems, and support for drives formatted in NTFS has always been either flakey or simply non-existent in Linux and macOS, so this move is certainly welcome.

We reported on Paragon's decision to bring its patch to the kernel back in **LXF272**, and earlier this year Linus Torvalds called on a new NTFS driver to be merged with the kernel (**https://bit.ly/lxf282linusntfs**). With the 5.15 kernel, it looks like he got his wish.

Paragon Software is bringing its previously commercial NTFS driver to the Linux kernel.

DISTROS

Ubuntu 14/16.04

Both LTS versions have their lifecycles extended to an impressive 10 years.

In some rather pleasing news for users of Ubuntu 14.04 and 16.04, Canonical has announced (**https://bit.ly/lxf282canonical**) that both LTS versions of the distro will have their lifecycles prolonged to 10 years, in what the company is calling the "Extended Security Maintenance (ESM) phase." The idea is that these will give companies that are still using the distros on their machines more time to plan and implement upgrades. As Nikos Mavrogiannopoulos, Product Manager at Canonical, says, "We are bringing an operating system lifecycle that lets organisations manage their infrastructure on their term." After the controversy over Red Hat cutting short the CentOS life cycle, it's good to see Canonical doing the opposite with Ubuntu.

Ubuntu 14.04 LTS 'Trusty Tahr' and 16.04 LTS 'Xenial Xerus' will now be supported for 10 years.

Distro watch

What's behind the free software sofa?

EMMABUNTUS DE4 1.00

The latest release of this Debian-based distro has now been released. Based on Debian 11 Bullseye, it comes with the Xfce and LXQt desktop environment. The project is designed to encourage humanitarian organisations to switch to Linux, enabling them to avoid paying licence fees and extending the lifespans of older PCs. There's a new-look theme, plus a renovated logo, and many proprietary apps such as *TeamViewer* and *Skype* have been replaced by open-source alternatives (*DWService* and *Jami*, respectively). Find out more at **https://bit.ly/lxf282emma**.

Emmabuntus is a distro for computers donated to humanitarian organisations.

SPARKYLINUX 2021.09

This lightweight Debian-based distro has a new version out, based on Debian's 'Bookworm' testing release. Repositories have been updated, there are new backgrounds and a login manager, and the Linux kernel is now 5.10.46 (though 5.14.6 & 5.15-rc1 are available in the Sparky unstable repos if you're feeling brave). If you're running the semi-rolling branch of SparkyLinux then you don't need to reinstall. To find out more about this version, head over to the release announcement at **https://bit.ly/lxf282sparky**.

A new update for SparkyLinux's semi-rolling release is now available to download.

EXTIX 21.9

ExTiX is a Deepin-based distro, and the latest version is now available to download. This new version lets you run the entire distro from RAM (provided your machine has enough, of course), and the installer has been changed to the Reborn version of the Deepin Installer. The kernel has been changed from 5.12.4-exton with kernel 5.14.2-exton. *VirtualBox* Guest Additions are now pre-installed, along with *Spotify* and *Skype*, and there are numerous other tweaks and changes. For the full list, head over to **https://bit.ly/lxf282extix**.

ExTiX can now be run completely in your PC's RAM for a major speed boost.

WHONIX 16

Whonix 16 is the latest major update for this Debian-based distro. As with other Debian-based Linux flavours, Whonix 16 is now based on Debian 11, along with all the new packages that introduces. A range of security features has also been added and expanded, including Hardened Malloc Kicksecure (HMK), Linux Kernel Runtime Guard (LKRG) and more. Whonix is designed to allow people to use the internet securely and privately, so these additions will definitely be welcome. Find out more at **https://bit.ly/lxf282whonix**.

Whonix? They'll never knownix, thanks to security features.

OPINION

STREAMY WINDOWS

Stéphane Cerveau
Senior Software Engineer, Collabora

"GStreamer is a powerful multimedia framework with over 30 libraries and more than 1,600 elements in 230 plugins providing a wide variety of functionality. This makes it possible to build a huge variety of applications, however it also makes it tricky to ship in a constrained device. Luckily, most applications only use a subset of this functionality, and up until now there wasn't an easy way to generate a build with just enough GStreamer for a specific application.

Thanks to major changes introduced recently to the GStreamer build system, you can now use it to generate a minimal, custom GStreamer build, tailored to a specific application, or set of apps.

When building embedded systems, there is often a need to do some sort of multimedia processing, and often resulted in needing to re-write an entirely custom application, which can be time consuming and error prone. Being able to use an existing, extensively debugged and optimized framework makes it possible to reduce development time, and with this new work, we can now make it fit in the available storage space.

These changes are already available upstream and should be available in the next major release (1.20) of GStreamer."

OPINION

HITTING THE M1

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

I continue to be shocked on a regular basis by the rapid rate of progress in the community that is porting Linux to the Apple M1 (Apple Silicon) Macs. As of this writing, and as reported by one of those involved (Alyssa Rosenzweig), "git status says the lines of code added for the Apple M1 is over 9,000". She is able to use the internal storage, HDMI, USB, Ethernet and much more as she works on enabling support for graphics (she is running GNOME on her M1 Mac).

At first glance, I (almost) wondered why anyone would bother. After all, support for virtualisation on that hardware is first rate and you can run a Linux guest VM faster than some 'bare metal' Windows (or Linux) laptops that are on the market. But while virtualisation might be an option for running Linux on top of macOS, it is of course never enough for the most dedicated developer. The most dedicated want to run Linux on the hardware just because they can. It's the same spirit that has driven many other efforts within the Linux community.

So I realise why these folks would 'bother' to do the work. Yet I still find myself conflicted between the amazing amount of work they have done – and the great progress that has been shown technically – and the potential difficulty that could be encountered by real-world users. The non-standard hardware uses a special bootloader, and requires various careful setups that will make dual-booting potentially awkward. That's not to say developers won't enjoy it, but just that it will be interesting to see that scale to many end distro users.

KERNEL WATCH

Jon Masters summarises the latest happenings in the Linux kernel, because someone has to.

Linus Torvalds announced Linux 5.15-rc1 (Release Candidate 1), noting that at just over 10,000 non-merge (non-administrative) commits, it is "in fact the smallest rc1 we have had in the 5.x series. We're usually hovering in the 12-14k commit range". He called out the new (writable) support for NTFSv3, as well as support for an in-kernel SMB (Windows network filesystem) driver that aims to play nice with the userspace Samba project and its utilities.

With the release of "-rc1" came the requisite closing of the "merge window" (period of time during which disruptive changes are allowed into the kernel) which had been "one of the messier merge windows". As Linus says, "Part of it was self-inflicted damage from me trying to enable -Werror much more aggressively". What he had done was attempt to enforce that all code warnings seen by the compiler would be treated as build-terminating error conditions.

In theory -Werror seems like a good idea. After all, who can't argue that a decrease in warnings isn't a good thing? And Linus has for years aimed to have no warnings in his own typical configs.

But, as a number of developers had pointed out, the kernel is large and complicated, and it is hard to find each and every minor warning that might stop a build. A compromise was reached in which that option was made conditional upon another (COMPILE_TEST) option. The goal is to improve the status quo without breaking many different CI systems.

Modern hardware frequently leverages workload-specific accelerators. These offload part of the work from the (general-purpose) CPU cores to special dedicated hardware that can often perform the same work orders of magnitude faster than if it were handled by regular software running on the CPU. With the slowing pace of Moore's Law, interest in accelerators has increased, but they were already critical components in order to reach high performance.

Examples of workload accelerators include the ML interference engines present in the latest generations of mobile phones. These can perform a very high level of machine learning on-device that would not be impossible with software alone. It's not just phones where accelerators are increasingly standard, but also in servers, workstations and laptops – all of which developers, as well as end customers, want to see running Linux. But there's a catch.

The catch is that there is often a lot of IP (intellectual property) locked up in the magic innards of these accelerators, some of which is enabled by special software that must run on the CPU to feed the accelerators. Upstream has begun to increasingly force a hard line in the sand: no open source userspace code to test an accelerator means no upstream acceptance of drivers for an accelerator. This had come to a head previously with the acceptance of a driver for the Habana Labs Tensor PRocessing Core (TPC) without an open userspace part.

To its credit, Habana released an open source LLVM-based compiler for its TPC, as well as various library code that should allow for a completely open (though not as performant as the closed source version) source stack.

This was well received, and seemed to further solidify the stance by developers against ever allowing another driver to go upstream without open userspace. LXF

» ONGOING DEVELOPMENT

Dave Jones reported a several-minute slowdown in boot time caused by newly cleaned-up support in the kernel for VPD (Vital Product Data) that can be read from various PCI devices.

Unfortunately, his rather common network adapter was an example of a device with a bogus VPD config, and this was used by Linus as a justification for reworking the code to avoid the boot slowdown. Meanwhile, Linus Torvalds noted that September 17th 2021 was "actually one of the core 30-year anniversay dates: 0.01 was uploaded Sept 17, 1991". He also noted that that release was never publicly announced, but "in many ways this is the true 30th anniversary date of the actual code". So there we have it!

SPORT

TECH

The Perfect Christmas Gift

from just £6

MUSIC

HOBBIES

SUBSCRIBE NOW

SAVE UP TO 41%

- *Give the gift that keeps on delivering*
- *Over 80 magazines to choose from*
- *Free delivery straight to your door*

www.magazinesdirect.com/xm21

Terms and conditions: Offer closes 31/12/2021. Savings are based on the newsstand cover price. Please allow up to six weeks for delivery of your first subscription issue in the UK, and up to 8 weeks for overseas. All gift subscriptions will start post-Christmas with the next available issue. Payment is non-refundable after the 14 day cancellation period unless exceptional circumstances apply. For full terms and conditions, visit www.magazinesdirect.com/terms. For enquiries please email help@magazinesdirect.com.

Answers

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

Neil Bothwick
says "mo Tux, mo problems"

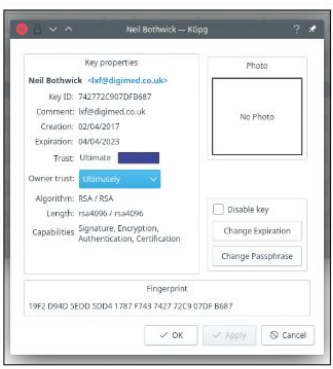
Key issue

I'm working on setting up for a switch from Windows 10 to Ubuntu (20.04 LTS) and getting all I would want or need from Windows to Ubuntu. I'm stuck on my PGP keys. I want to export them from Windows to Ubuntu. I find no easy way to do it. I need to export both public and private keys from Windows (I've tried with *Kleopatra* with no solution that works for me yet) and import the same into Ubuntu and into *Thunderbird*.
Steve Cox

A There are two ways to do this. The first is to copy your entire GPG dataset from Windows to Linux. On Windows 10 this is stored at **C:\Users\username\AppData\Roaming\gnupg**. Copy this directory to **~/.gnupg** and make sure everything is owned and only readable by your user. You can do this with:

```
$ chown -r $USER ~/.gnupg
$ chmod -R o-rwx,g-rwx ~/.gnupg
```

If you just want the keys, rather than everything, you can export them in Windows and import them in Linux. The Windows side can be done from *Kleopatra*. Right-click the key and select Export. Then right-click again and select Backup Private Key. This will give you two .asc files that you need to copy to your Linux system. Then you can import them with



GnuPG keys can be exported in Windows for import in Linux, and vice versa.

```
$ gnupg --import username_ID_public.asc
$ gnupg --import username_ID_SECRET.asc
```

The final step is to add ultimate trust to your keys. Ultimate trust is rarely used, but these are your keys, created by you, so it applies here. Run

```
$ gpg --edit-key your@email.address
```

It will list your key and show a prompt – you're in the *gpg* editor now. Run the command `trust` and it'll give you choices for levels of trust to apply. Select the "I trust ultimately" option, confirm it and type `quit` to exit the editor. Now you have the same keys available on your Windows and Linux systems.

For the sake of completeness, if you ever need to go the other way, you can export your keys from Linux with the following commands:

```
$ gpg --export-secret-keys -a yourkeyID
>gpg_private_key.asc
$ gpg --export -a yourkeyID >gpg_public_
key.asc
```

Then you can transfer them to Windows and import, either in *Kleopatra* or using the *gpg* command line client as in Linux.

One final word of warning: be careful with the exported files. We'd advise saving them to a USB stick when exporting, import directly from there and then format the stick after use, or even zero it with *dd*. You can't be too careful with privacy.

Shutdown Shenanigans

We have a Kubuntu Linux machine that I have set up so several people here can test a Linux system. This enables them to see if they would like to use Linux for their daily work.

This is working out fine, except for the issue that each user is able to shut down the machine at the end of their testing. They aren't logging off but shutting down the machine. I find this not to be acceptable, but I don't know of any way to stop this from happening. I would like to restrict these users for being able to shut down the system, but I haven't figured out a resolution after trying a number of things – nothing works.

Is there a way to define a general user from only logging off and not being able to shut down the system. We're only using one physical terminal for the Linux system at this time. My user ID should be the only super user that's able to shut things down.
Paul Burnham

A How you approach this depends on the type of behaviour you're trying to prevent. If you're concerned about someone deliberately shutting the computer down when they know they shouldn't, you'll need a more robust solution, possibly using *SELinux*. However, it appears you're trying to guard against careless behaviour, so the first thing we'd do is take the low-tech approach: put a sticker on each computers saying "Please don't shut me down".

Enforcing that turned out to be harder than we thought. KDE has an option to not show shutdown options, in the Session section of System Settings. However, it made no difference. Then we looked at *SDDM*, which KDE uses to do the actual shutting down. This has a configuration option to change the command that's used to shut down the computer. It too didn't work. However or wherever it was set, *SDDM* insisted on calling systemd's `systemctl poweroff`.

So we looked at that and found a solution. First we need to modify the shutdown service by creating the file **/etc/systemd/system/systemd-poweroff.service.d/00-local.conf** containing:

```
[Unit]
Requires=checkshutdown.service
After=checkshutdown.service
```

This makes systemd run this service before the `poweroff`. Now create the service file at **/etc/systemd/system/checkshutdown.service** containing

```
[Unit]
Before=shutdown.target umount.target
final.target
[Service]
Type=oneshot
ExecStart=/usr/local/bin/checkshutdown.
sh
```

The 'Before' line ensures that it runs

before any of the other services that `poweroff` calls – we don't want those to run unless we're actually shutting down. `ExecStart` runs the given script, which you should create containing

```
#!/bin/sh
MY_UID="yourusername"
if users | grep -q $MY_UID; then
 true
else
 false
fi
```

This script simply succeeds if your user is logged in and fails otherwise. Since `poweroff` now depends on this, the shutdown will be aborted before it starts if you're not logged in. Finally, run:

```
$ sudo systemctl daemon-reload
```

This will make systemd read the new or changed files. The example here is very basic because it doesn't check exact user names, only for a string match, so for example, if your username is `johnsmith` then `theotherjohnsmith` will also be able to shut the machine down.

It would also be nice to add something to the false section to inform the user that they cannot shutdown, possibly using *libnotify* – this just ignores their request. We will leave that as an exercise for you.

Q Lost forever

I can't seem to delete a locked folder named lost+found that was created by my back-up utility and placed on my back-up hard drive. I've right-clicked the folder and tried to give myself permission to, but there is no option.

Michael Piziak

A The **lost+found** folder is created by the ext4 (or ext2/3) filesystem in use on your hard drive. When the filesystem is created, a **lost+found** folder is created in the root of the filesystem. The folder is used by *e2fsck*, the filesystem recovery tool, to store "orphaned" files in the case of filesystem corruption. You can delete it and it does no harm because the folder will be recreated whenever it's needed or you run *fsck*. On the other hand, there's not much point in deleting it because it's empty and will be recreated whenever it's needed or you run *fsck*.

If you're a neat freak, by all means delete it, but you'll have to do this as root. You were unable to give yourself permission to delete it because it's in the filesystem root, so you'd need to own the root of the filesystem to delete it. Furthermore, there would be no point in applying permissions to files if a user that didn't already own them could change them. Note that with Linux filesystems, the filesystem itself has the ownership property, and so changing the mount point has no effect.

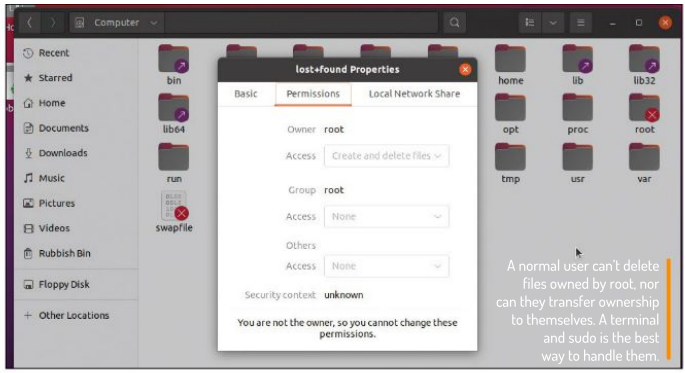
You can delete it from the terminal with the `rmdir` command, but it has to be done as root:

```
$ sudo rmdir /media/hd/lost+found
```

and adjusting the mount point to suit your setup. The `rmdir` command only removes empty directories, so it's safe to use here as **lost+found** should be empty. If `rmdir` fails with a `Directory not empty` error, there are files in here and you need to investigate where they came from, and also check the integrity of the filesystem with *fsck*. This is done on an unmounted filesystem, with a command like:

```
$ sudo fsck -N /dev/sdb1
```

The -N flag instructs *fsck* not to try to fix anything – just report any problems. Run it again without -N to try to fix any that are found.



Q Growing logs

I was trying to free up disk space and noticed this command to delete all journal entries older than two days:

```
$ sudo journalctl --vacuum-time=2d
```

It resulted in:

```
Vacuuming done, freed 1.1G of archived
journals from /var/log/journal/....
```

Where is the best place for this to be run every day? Either a bash script at startup/boot or a cron job?

Jimmy Lawson

A The best way to run a command every day is to put it in a script in **/etc/cron.daily**. This runs the command regularly, instead of depending on when you boot or log into the desktop. This is if you want the command run as root, otherwise put it in your user's crontab. However, this isn't the ideal way to deal

» A QUICK REFERENCE TO... FILESYSTEM LABELS

Back in the not-so-good old days when computers had only one hard drive and 640K was enough for anyone, it was acceptable to refer to partitions by their device nodes: **/dev/hda1** and so on. As more devices were connected to our computers, this because a problem: for example, the partition that is **/dev/sdb2** today could end up as **/dev/sdc2** tomorrow.

To avoid this issue, many distros began to use filesystem UUIDs to identify what was mounted where. This works because UUIDs are 16-digit hexadecimal strings that are (almost) guaranteed to be unique, but it was sorely lacking in the human-friendliness department, meaning that most distros had two lines per device in **/etc/fstab**, one with the UUID and a comment line explaining what the heck this meant.

You can avoid this obfuscation by using filesystem labels. You can apply a label to a filesystem when creating it, or later on. For example, with an ext4 filesystem, you can add a label with:

```
$ e2label /dev/sda1 myroot
```

Include the hostname in the label and you're back to the same level of uniqueness you get with UUIDs, except the label tells you something – even when the hard drive is removed from its computer. Now you can replace those hideous lines in **/etc/fstab** with something as clear as:

```
LABEL=myroot / ext4 defaults...
```

with the space occupied by the systemd journal, because it's capable of managing the space it occupies on its own, without relying on external scripts.

Systemd is already watching and controlling the amount of space the journal uses, you just set a much tighter limit when you ran *journalctl* with the `--vacuum-time` option (there's also a corresponding `--vacuum-size` argument to reduce the journal to the given size). The size of the journal is limited in two ways: to not exceed 10 per cent of the total filesystem size and to leave at least 15 per cent space free in the filesystem. These are controlled by two settings in **/etc/systemd/journald.conf**: `SystemMaxUse` and `SystemKeepFree`. Set one or both of these to your preferred values to further limit the space used by the journal. Irrespective of these settings, the journal will never exceed 4GB.

You can also set the compress option in **journald.conf** to reduce the space taken by larger entries, although this could affect journal performance.

It's best if you don't try to limit the space used by the journal too aggressively, you never know when you may need to refer back when investigating a problem – for example, to compare how something behaves now compared with before some problem arose. After all, you can still use the `--vacuum-size` option manually if you're ever in the position where you need all the space you can get.

Q Swapping thumbs

I'm trying to install Mint to a USB thumb drive. I have MINT bootable, from an ISO, on a thumb drive but am trying to get past that to a full installation on there. I specified partitions as:
EFI 500 MB
/boot 1GB
/swap 2GB
/ the remainder, around 14GB

But the MINT installer then tells me that my internal NVMe drive will be formatted as swap! Whoa! For what it's worth, the same happens when I try the Xubuntu installer. That would probably mess up my main-man PC Linux, which would be a tragedy.

How do I get /swap on thumb during the installation process?
Roger

A The Mint live CD, in common with some others, looks for existing swap partitions when booting and mounts those. It doesn't touch any filesystems for data, only swap, which makes sense as you then automatically have swap space when booting from the live distro. It can do this because swap unlike, say, **/home** can be freely used between different distros.

Because you already have a swap space mounted, the Mint installer then takes this as the default choice for the new installation (because swap can be shared between distros). It should be sufficient to unmount the swap space before running the installer. You can see which partition is being used for swap with this command

```
$ sudo swapon -s
```

Then you can unmount it with:

```
$ sudo swapoff /dev/nvme0nXpY
```

using the device name returned by `swapon`. If that's not enough to stop the installer wanting to use your hard disk, and if it won't let you change, you have a couple of options. The first is to manually partition the drive as you want, create a swap space on the relevant partition and then mount it:

```
$ sudo mkswap --label usbswap /dev/sdb3
$ sudo swapon -L usbswap
```

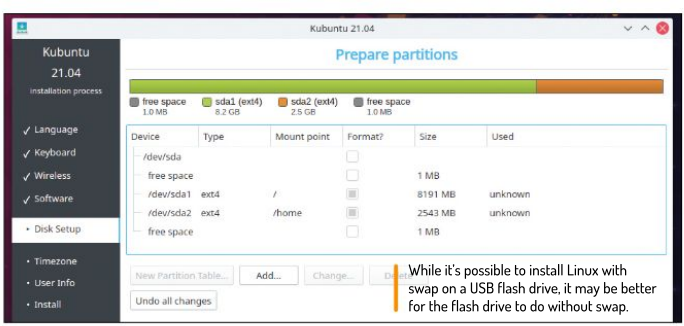
Using a label is advisable – USB sticks can come up with different devices names each time they're used. Now that your active swap is on the stick, the installer should be more amenable to your wishes.

The other options is to not use a swap partition at all, just tell the installer you don't want swap. Swapping to a flash drive is probably not that good an idea because it could reduce its lifespan. If at any time you find you need swap space, you can create a swap file, like this:

```
$ sudo fallocate -l 2G /path/to/swapfile
$ sudo chmod 600 /path/to/swapfile
$ sudo mkswap /path/to/swapfile
```

Then use it as swap space with:

```
$ sudo swapon /path/to/swapfile
```



Q Partial upgrades

When I do `sudo apt-get upgrade` **I get the following errors:**

```
The following packages have been kept back:
gnome-shell-extension-desktop-icons-ng
ubuntu-advantage-tools
```

How do you fix this on Ubuntu 21.04?
Robert

A This isn't an error, only an informational message that would normally disappear in due course. What is happening is that some software you've installed has a dependency on the current version of one of these packages that's not satisfied by the newer release. This is simply *apt* informing you that in order to keep all of your software working, it's not upgrading those packages at present.

This only becomes a problem if something requires the newer version while a different package requires the older version, so you get conflicting dependencies that can't be resolved. This shouldn't happen in a carefully managed package repository, and if it does it's generally fixed quickly, but it can happen when using PPAs that don't necessarily have the same level of quality control.

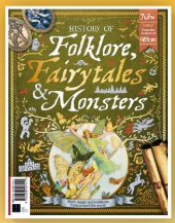
Running `apt-get` with the `--verbose` flag should give you a better idea of what's contributing to this situations. If it persists it would be worth filing a report on the Ubuntu bug tracker. LXF

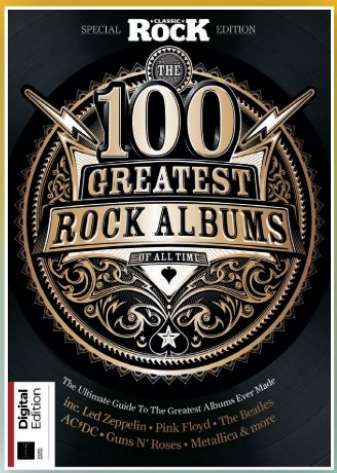
GET HELP NOW!

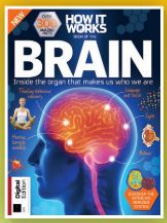
We'd love to try and answer any questions you send to **lxf.answers@futurenet.com**, no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (although many suspect Jonni is a robot), so it's important that you include as much information as you can. If something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

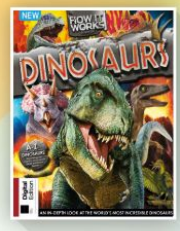
If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us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

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



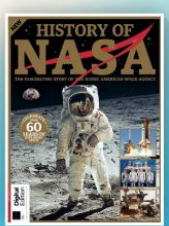






DISCOVER OUR GREAT BOOKAZINES

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



Follow us on Instagram @futurebookazines

FUTURE

www.magazinesdirect.com

Magazines, back issues & bookazines.

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 **lxf.letters@futurenet.com**.

Zero trust

I've only skimmed through portions of this US goverment draft document (www.cisa.gov/publication/zero-trust-maturity-model) so far. Some things may help, but we won't know for a year or more. The metric will be ransomware and malware in general, going down in occurrence. I have my doubts.
Ed Scott

Neil says...
I'm not a security expert, nor am I an expert on current US government IT security practices. However, having had a read through of that, much of it feels like solid good advice and security practice, I'd say some of the points are how systems are run here at *Linux Format Towers*. We're all on single-sign on, two-factor authentication, segmented networks, rolling password changes, strong device audits. If these aren't implemented by US government already I'd certainly be worried as they feel like a minimum required by even small-sized companies and, to be honest, individuals.

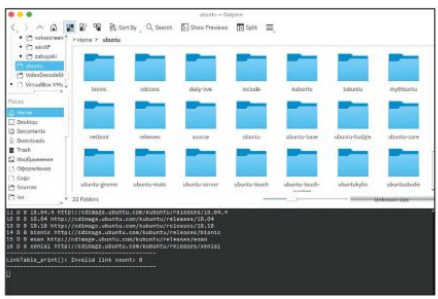
As mentioned in the report some changes (I'd say TFA and strong device auditing) require cultural change. There was a level of pushback by our US colleagues when told they can't use their own devices any longer. Not sure this a US-specific thing, or just a media-based attitude. It's an interesting but essential development.

You're so hot

I'm the author of *HTTPDirFS*, which was featured in the July 2020 issue of *Linux Format*. I have only discovered this two days ago. I bought a second-hand copy of the magazine off eBay! Anyway, I have an even more popular repository that solves a lot of people's pressing problems. I'm very proud of it. I'm just wondering if you could feature it in *Hotpicks*? The link to the repository is here: https://github.com/fangfufu/Linux-Fake-Background-Webcam
Fufu

Alexander says...
That's a nice coincidence since I was playing lately with a similar project: **https://github.com/jashandeep-sohi/webcam-filters**. I couldn't make it work in my VM despite the correct webcam passthrough, so I decided to postpone the tests and come back to it later on the bare hardware setup. I mean that, otherwise we would have had that software featured in **LXF282**.

Y'know, projects involving OpenCV+Python often misbehave when launched on a Linux distribution that is different from the project developer's one. But I'll definitely give the Fang's version a try. It's way more popular than the project I've already tried. If it is good, then I'll include it in the next *Hotpicks* series. So the answer is yes.



Hotpicks covers some outstanding software, so please do suggest anything you find!

It's a RAID

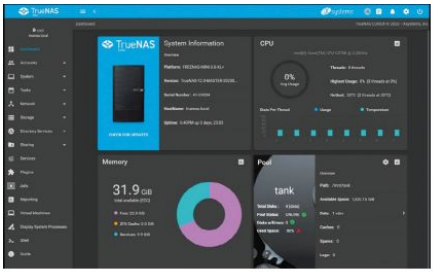
I recently had a RAID 6 disaster and learnt that a hot swap capability is essential so that you don't have to demolish your server to swap the most tucked-away disk. I was also reminded why I keep backups.
While researching RAID I reviewed my options such as FreeNAS and RAID, and read about combining LVM and *MDADM* aka Linux software RAID. I was also surprised to read that LVM uses at least some of *MDADM* under the hood.

I then looked at BTRFS and found ZFS. OpenZFS on Ubuntu Server looks very attractive as a RAID and LVM alternative.

How about some articles explaining whether, when and how to cobble together RAID and LVM, including whether to use one on top of the other and why, and how to use ZFS and its pros and cons against the other two.

Finally, although I've been using Linux for about 12 years, I prefer to use a GUI when possible. Is there is a one-stop GUI for managing this stuff?
Mike Sapsard

Helpdex



CREDIT: truenas.com

Sometimes it's easier to just go with a ready-packed solution. Why take chances with your data?

Neil says...

Your request has arrived with pretty good timing because coming up in the magazine is a group test on file systems, which should give you an overview of what's available. We'll also follow this with a tutorial on setting up Btrfs and examining its capabilities.

If you have a look at the archives we've covered ZFS in **LXF273** and **LXF227**. While it's excellent at what it does, it's a high-end server solution and expects a lot of memory (8+GB) to be available. If you're after a one-stop solution have you looked at TrueNAS/FreeNAS? It uses ZFS and offers an all-in-one storage solution based on BSD? I popped *Webmin* on my home server so I could keep an eye on the RAID as one GUI option. But yes, general trends are moving away from raw RAID and to logic volume management.

Niche but nice

Any chance of an article on installing CP/M on a Raspberry Pi 4?

Andrew Deacon

Neil says...

Wow that's pretty niche (which is an amazing thing for an editor of a Linux magazine to say), but I think we're getting to the point where we've hit up every non-gaming classic platform, so if Les Pounder is game then he can add it to the list! And I'm sure you've come across this project during your travels: **https://github.com/rsta2/cpmemu**.

LETTER OF THE MONTH

You hypocrites

The article about 'hypocrite commits' (LXF277) reminded me of an incident that happened 20 years ago. (To save space and to avoid "Mike-splaining" I'll assume everyone will either remember, or can look up, information about the American TV series *Survivor*).

Around August 2001, newspapers and media networks went wild over reports that an unknown 'hacker' had discovered the identity of the winner of that season's *Survivor*. Never mind that they couldn't identify this hacker, nor verify the claim. And never mind that there were far fewer 'netizens' than there are today, the average person couldn't verify it, either. They said that Jervis won, so it must be true. So, two weeks later, when he was 'voted off,' the same news people waved their virtual – as opposed to virtuous – fingers in our faces and announced "See? You can't trust the internet!" In a similar fashion, the prank pulled by the University of Minnesota sounds like an attempt to say, "You can't trust the Open Source Movement!"

Mike Neary

Neil says...

Who knows what the true motivation was. I could see an honest project like this unfortunately going down this lazy route. It's a missed opportunity to improve code quality. If the project had worked from the beginning with the Linux leadership team with a planned and traceable series of code submissions, there would have been an opportunity to see if the code review process could be improved and if there were any obvious weakness. Instead, we just get drama!

CREDIT: Michael Specht, CC BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Sanco_8001.jpg

Haven't we all wanted to run an eight-bit 1970's operating system at some point?

shane_collinge@yahoo.com

GET THE LATEST LINUX UPDATES, TUTORIALS AND EXPERT ADVICE

Collating some of the best content from 12 months of the leading Linux magazine, the latest Linux Format Annual offers in-depth features, fun step-by-step guides, Raspberry Pi projects, security tips and much more!

ON SALE NOW

FUTURE

Ordering is easy. Go online at:
magazinesdirect.com
Or get it from selected supermarkets & newsagents

FireCuda 530 2TB

Seagate puts the SSD market on notice, says **Chris Szewczyk**.

SPECS

Capacity: 2TB
Format: M.2 2280 (80mm)
Interface: PCIe 4.0 x4, NVMe v
Logic: Phison PS5018-E18
Memory: Micron 176L TLC
Sequential read/write: 7,300/6,900 MB/s
Endurance: 2,550TBW
Security: N/A
Warranty: Five years

We've been waiting in anticipation for the release of Seagate's FireCuda 530 series and we can happily state that it's definitely time to add Seagate to the list of manufacturers capable of making top-tier SSDs.

The FireCuda 530 is a 2280 M.2 drive. It combines brand-new Micron 176-Layer TLC NAND with a Phison PS5018-E18 controller. This gives it a rated sequential read and write speed of 7,300/6,900MB/s, respectively, which is pushing the limits of a PCIe 4.0 x4 interface. The 4TB model has the same rating while the smaller 1TB and 500GB drives are rated for 7,300/6,000MB/s and 7,000/3,000MB/s, respectively. Another key point of note are its two versions: one with a heatsink, and the other without; we're testing the heatsink-less version.

A key characteristics of the range is its stunning endurance ratings. The 2TB model comes with a massive 2,550TBW endurance (total bytes written), while the 4TB ups this to 5,100 TBW. These make the likes of the Samsung 980 Pro and WD_Black SN850 look like budget bin cheapies in comparison. The lack of AES256 encryption is a notable omission, but Seagate would likely say that drive encryption is unnecessary on a 'gaming' SSD.

The FireCuda 530 pictured with and without its heatsink.

Running with the big boys

So, how does it perform? If Seagate hopes to sell a drive that costs more than the established SN850 and 980 Pro then it has to deliver, and it does. The latest Phison E18 drives excel at sequential read and write tasks, but tend to trail in random performance. The FireCuda 530 either matches or beats the big boys of the storage world, and when you add its leading sequential performance and endurance rating, the Seagate 530 is at least the equal of any consumer SSD on the market.

We record temperatures throughout our testing making note of the peaks. The drive is left to run 'naked' with a Noctua fan over the area to keep things cool. The 2TB FireCuda 530 reached a peak of 71°C, which makes it a toasty drive, though we never saw any throttling during our extended testing. You'll either want good airflow or an effective motherboard heatsink to keep things in check with this model.

Say hello to the fastest SSD on the market today.

As a major player in the storage industry, you'd expect Seagate likely wrestled with the decision to hold back on releasing its best SSDs until it could include 176L NAND and steal the headlines. For us, the decision was absolutely worth it. Class-leading sequential performance and a stunning endurance rating is joined by excellent random read and write performance.

All up this elevates the Seagate FireCuda 530 to the head of the pack. It'll make a great boot drive, it'll hold a substantial software library or you could use it as a scratch disk for large data sets that need to be moved frequently. Feel free to thrash it with any kind of task.

Seagate knows it has a strong competitor on its hands, and that means prices are steep. Probably too steep. There are strong competitors for less money – in fact, the Seagate 530 2TB appears to be the most expensive of all consumer 2TB SSDs. When you combine stunning endurance ratings with awesome performance, then it's no surprise the FireCuda 530 comes at a premium. Pricing is always volatile though, especially in the current climate.

But Seagate has well and truly shaken up the top end of the SSD market. Samsung and WD now have some serious competition. If you're prepared to pay for the privilege, buying a Seagate FireCuda 530 will reward you with top-tier performance and particularly reliability ratings that no competitor can match. LXF

VERDICT

DEVELOPER: Seagate
WEB: www.segate.com/firecuda
PRICE: £372 2TB (4TB £800, 1TB £185)

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

Seagate's 2TB FireCuda 530 combines leading performance with best-in-class reliability to take the SSD crown.

» Rating 9/10

Asus Tinker Board 2S

This is the little board that couldn't, says a disappointed **Les Pounder** as he tries a board that's definitely no Raspberry Pi.

IN BRIEF

SoC: Rockchip RK3399
CPU: dual-core Cortex-A72 @ 2.0GHz, quad-core Cortex-A53 @ 1.5GHz
GPU: Mali-T860 MP4 GPU @ 800MHz
RAM: 2GB (4GB)
Storage: 16GB eMMC, Micro SD
Display: 1x HDMI, 1x USB Type-C, 1x 22-pin MIPI DSI
Comms: 1x Gigabit LAN, 1x 802.11ac, Bluetooth 5.0
Audio: 1x HDMI audio, 1x S/PDIF via GPIO, 1x PCM/I2S GPIO
USB: 3x USB 3.2 Gen1 Type-A, 1x USB 3.2 Gen1 Type-C
GPIO: 40-pin headers including up to: 28x GPIO, 2x SPI bus, 2x I2C bus, 2x UART, 3x PWM, 1x PCM/I2S, 1x S/PDIF TX, 2x 5V power, 2x 3.3V power, 8x Ground
Power: 12~19V DC
Size: 85x56mm

Designed to outmatch the Raspberry Pi 4, the Asus Tinkerboard 2S has a few features that look good on paper compared to the Pi, including a six-core CPU and 16GB of on-board storage.

The Asus Tinker Board 2S has a big.LITTLE CPU configuration, with two 2GHz Arm Cortex A72 cores and four 1.5GHz Arm Cortex A53 cores. When we need the extra power, the A72s kick in, but for general tasks the A53s get the job done. That's compared to four, equal 1.5GHz Cortex A72 cores on the Raspberry Pi 4.

The default operating system for The Asus Tinker Board 2S is Tinker OS, a fork of Debian 10 "Buster", which uses LXDE as the desktop environment. Tinker OS is a functional and responsive OS, but it does feel like an old spin of Raspbian.

YouTube playback at 720p is impressive and quite responsive. Moving from windowed to full-screen playback was slow, but faster than a Raspberry Pi 4. We then tested video at 720p 60Hz and 1080p 60Hz and here's where we started to see the worst results. There were lots of dropped frames in windowed and fullscreen, and at 1080p 60Hz the video buffered and stalled.

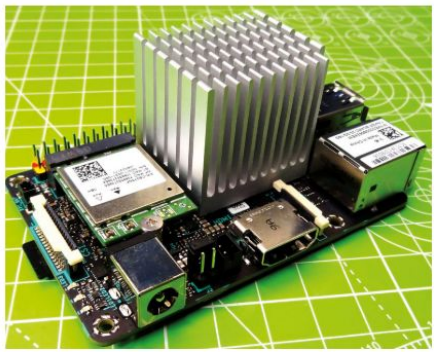
GPIO stumbling blocks

There's a 40-pin GPIO, which mimics that of the Raspberry Pi, but we have a few issues with it. Chiefly there's no compatible Python 2/3 library that we can use with the GPIO. For the previous Asus Tinker Board there was ASUS.GPIO, which was a fork of the RPi.GPIO library, but alas that library doesn't work with the 2S. We reached out to Asus for support and were advised that Asus is working on "additional information" on how to get the GPIO working with Python.

We did manage some basic GPIO access using *WiringPi*, specifically the GPIO command. Using a simple BASH script we created a simple button input, LED output project that worked well, but we still longed for a useful Python library.

Dominating the Asus Tinker Board 2S is an aluminium heat sink, which keeps the CPU cool, and unfortunately blocks HAT access to the GPIO. Wi-Fi and Bluetooth connectivity requires two external antennas that are supplied, without these the Tinker Board will struggle to connect to even the strongest access point.

Asus needs to deliver the software for people to use.



Although designed to be like the Raspberry Pi, it's not quite the same.

The onboard 16GB eMMC of the Tinker Board 2S is a decent feature. We copied a 2.8GB file from a SATA USB 3 drive and recorded just under 51 seconds to copy it to the eMMC. The same test came in at 52 seconds using a 16GB A1 class micro SD card. The eMMC is nice to have, but we can make do with a good-quality micro SD card. A jumper enables you to switch booting between microSD and eMMC.

As it stands, the Asus Tinker Board 2S is fine for desktop tools and as a home server. It has enough power to perform those tasks admirably. As a maker hobby board, the Tinker Board 2S is crippled by the inadequate GPIO access. When the Python GPIO library arrives this may change, but don't expect the same level of quality as the Raspberry Pi.

For $129 we expected better, and when we consider the Raspberry Pi 4 is $75 and comes with everything we need for a desktop and maker experience, the Asus Tinker Board 2S can't compete on price, or challenge the dominance of the Raspberry Pi ecosystem. LXF

VERDICT

DEVELOPER: ASUS
WEB: https://tinker-board.asus.com
PRICE: $129

FEATURES	**6/10**	**EASE OF USE**	**4/10**
PERFORMANCE	**6/10**	**VALUE**	**5/10**

The Asus Tinker Board 2S is too expensive for what is just a badly supported Pi alternative. Buy a Pi 4 instead.

» Rating 5/10

Zorin OS 16 Core

A way to move operating systems, or a desktop with an identity crisis? **Neil Bothwick** pretends to be a Windows lover while he tries Zorin OS.

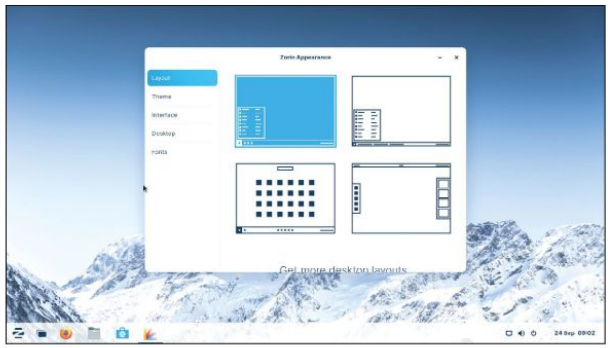
IN BRIEF

Zorin OS is a distribution aimed at those making the journey from Windows or macOS to Linux. To this end it can be made to look like one of those systems, at least as far as theming allows. It is more than just a paint job though; Zorin OS has several additional features that make upgrading from Windows to Linux less traumatic.

SPECS

CPU: 1GHz 64-bit
Memory: 2GB
HDD: 15GB Core, 30GB Pro
Build: x86 64-bit only

There are several Linux distributions that aim to provide an easy introduction to Linux for those coming from Windows. They may do this by providing a Windows-like appearance, or go deeper. Zorin OS falls into the latter category. It's available in three versions: Core, Lite and Pro. Core, the version we have here, is a free distribution, in both senses of the word. Lite is a lightweight version designed for older computers and available in 32-bit as well as 64-bit flavours. The Pro version is a paid product, currently costing £46.80.



Pick a desktop, any desktop! Zorin Appearance makes it easy to have your desktop emulate the look of a different OS.

Zorin OS is based on Ubuntu, which is clear when you run the installer. It follows the same process as the standard Ubuntu installer, even though it looks different. The same is true of the Zorin Gnome desktop – it has its own look, a clean one that I found very pleasant to work with. In fact, there are several desktop appearances to choose from; you switch between them by selecting Zorin Appearance from the main menu.

These choices give a clue to Zorin's target audience, those coming from Windows or macOS. It is not that the desktop layouts try to emulate those other operating systems, but they do provide a more familiar introduction, or you can go for the standard Gnome shell layout as used by Ubuntu. There are further layouts available with the Pro edition, along with extra software. However, we didn't have access to a Pro version and the website is quite vague about the extras.

Added Wine

Zorin tries to handle the needs of Windows users as best it can, including being able to install (some) Windows programs. When you run a Windows EXE installer, it offers to install it via "Windows App Support". Unfortunately, the installation of that software then fails with a cryptic message about unmet dependencies. Installing Windows App Support from the software manager worked though, although this is not something a first-time Linux user would be expected to know.

Windows app support is basically *Wine* and *PlayOnLinux*, packaged so that Zorin can run Windows apps whenever possible. It is a good idea to help Windows users start with Linux without having to abandon all the software they know, but it needs some work. One nice touch is that if you try to install something for which a native Linux version exists, such as *Thunderbird*, Zorin will offer to install it from the *Software Manager*. Another addition to the standard Ubuntu fare is *Zorin Connect*, a way of keeping your computer and Android device in sync, including displaying notifications, controlling music playback and even using your phone as a slideshow remote.

When a distro is aimed at new Linux users, effective hand-holding and documentation is important. The website contains some guides; one on installing Zorin and an excellent, comprehensive guide on installing software from various sources. The latter is important as this is an area that confuses those coming from Windows. It is just a pity that there is not more documentation, especially as the quality of what is there is so high.

Because it is based on Ubuntu, and can run in a Windows-like desktop or more vanilla Gnome, Zorin OS is not only for beginners. It is a distro that new users can start with, but they can also grow with it, getting rid of the Windows theme when it is time to remove the training wheels. LXF

VERDICT

DEVELOPER: Zorin Group
WEB: https://zorin.com/os
LICENCE: Various open source

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

Zorin makes a good job of easing the transition to Linux. The *Wine* feature needs some work, but *Wine* is an imperfect solution anyway.

» **Rating 9/10**

Rescatux 0.73

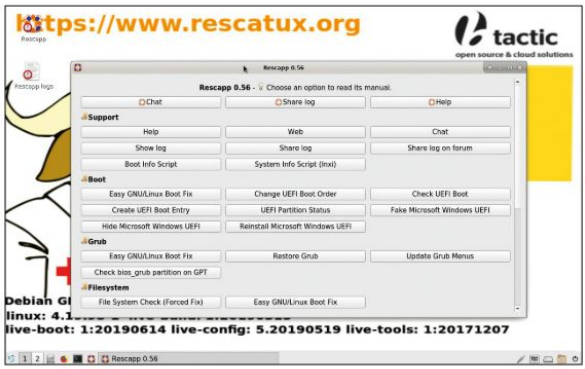
With its fairly descriptive name for a distro, **Neil Bothwick** finds out in just what situations it can rescue our little penguin.

IN BRIEF

Rescatux is a live distribution the provides an easier way to solve some of the more tricky problems that can beset our computers. The one image supports both 32-bit and 64-bit hardware and can be used to recover from most of the common situations that can prevent your computer booting, both for Linux and Windows.

Among all the problems that our readers write to *Answers* for help with, by far the most popular (?) are booting issues. The vast majority of those relate to *GRUB*. Unlike its predecessor, *GRUB2* is a complex beast. This is good for distro makers as it allows boot setup to be automated, but it can make life difficult when things go wrong.

Things going wrong are often caused by attempts to set up multi-booting, especially if you have Windows and more than one Linux installation. Web searches for solutions often result in arcane command line or *GRUB* prompt invocations that may as well have originated in Hogwarts for all the sense they make. What we need is a simple way to carry out repairs and maintenance of the boot options, and Rescatux is it.



More than a windowful of solutions to some of the awkward problems that can stop your computer in its tracks.

When you boot the Rescatux live CD or USB stick, you are presented with *Rescapp*, basically a window full of buttons. Each button carries out a particular task, such as regenerating your *GRUB* menu, altering your UEFI boot options, checking partition tables and filesystems and much more.

There is a lot more to Rescatux than fixing the odd *GRUB* glitch. There are several problems that cannot be fixed from a running system, either because it will not boot or because you need the filesystems to be unmounted; running *fsck* on the root filesystem of your distro is an example of the latter. Examples of the former could be a forgotten password – it can fix that too. What if you can boot but you cannot use *sudo* to perform administrative tasks? You guessed it, Rescatux can rewrite your **sudoers** file to let the specified user perform root tasks, although it may well have been that ability that got you into trouble in the first place!

Read before running

All of this is pretty low-level stuff, stuff you shouldn't be messing with without having some clue as to what is going on. Rescatux is on top of that because none of the buttons actually runs a command. Instead they each open the documentation on what that option does, which you can read before pressing the Run button to actually perform the selected task. Whether you read the documentation is up to you, but if things do go wrong, you have no one to blame but yourself.

There are more advanced options available, ones that you would otherwise run manually from a live distro like SystemRescue. Rescatux is not limited to dealing with Linux problems either, it can help sort out Windows problems from forgotten user passwords to dealing with booting and UEFI issues. There is also a complete desktop, with browser, so you can search the web for further help should your problem demand it.

At this point, you may be asking if there is anything Rescatux cannot do. Well, this is not a hacking tool; you won't be able to crack encrypted filesystems or containers with it – resetting passwords as mentioned is something any distro can do with root privileges.

Anything that Rescatux can do can also be done with a decent rescue distro like SystemRescue. In most cases it can also be done with a general distro live disc. Where Rescatux scores is in the way everything is packaged into a straightforward frontend that takes care of the details, and those arcane incantations, while still giving you all the information you need to know before proceeding.

Now all we need is a live distro to fix the second largest cause of problems: printing, believe it or not... LXF

VERDICT

DEVELOPER: Adrian Gibanel Lopez
WEB: https://rescatux.org
LICENCE: Various open source

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

We hope you won't need it, but keep a copy of Rescatux on a USB stick just in case.

» Rating 9/10

Deepin 20.2.3

Deepin is an independent distro originating in China. **Neil Bothwick** tries it to see what he can take away from the experience.

IN BRIEF

This is an independent Linux distro with its own desktop environment and a suite of applications that have the same look and feel. The desktop is attractive and easy to work with, and the environment is well integrated. The provided documentation is of a high standard.

SPECS

Minimum
CPU: Pentium 4 2GHz 64-bit
Memory: 2GB
HDD: 25GB
Build: x86 64-bit only

Most Linux distributions are derivatives of other distros, and usually quite obviously so. The most popular base is Ubuntu, which is itself based on Debian. It's unusual to find a distro with its own, separate identity, but Deepin is one such distro.

In some ways, it goes back to basics. The ISO image is an old-school installer system; there's no live environment to try it out and so the distro has to be installed. You can use a virtual machine for this, but it does run better when installed on real hardware.

The included documentation is excellent, very informative and easy to navigate.

When starting the installer, you're asked to agree to the licence conditions. This is unfamiliar territory for most Linux users. The Deepin licence is quite wordy, but basically makes the software open source while protecting the developer's trademarks and identity.

Once installed, Deepin boots to an attractive-looking desktop, using a customised Gnome setup. Many aspects of Deepin are heavily customised, which makes a refreshing change to the sameness of many distributions. Granted, most distros consist of mainly the same components – a kernel, a desktop environment and applications such as web browsers, media players and office suites – but Deepin tries to look and feel different, even if under the skin not much has changed.

In most cases, customisation is a matter of taste, although we really liked Deepin's desktop appearance. However, changing the functioning of software in less-than-trivial ways is another matter. One of the first things many people do with a computer is fire up a browser session and search for something that's piqued their interest. The browser is Deepin's own, although it's based on Google's *Chromium* code so it still uses **chrome://** URLs for settings.

The problem for most users is that the default search engine is Chinese, as are the only two other choices, so most results are Chinese. Yes, you can add search engines, but should you have to search for how to add Google search to a Google-based browser? Deepin is Chinese, but it's released for an international audience and language choices are given during installation, so why are they not respected here?

One disadvantage of using a distro which isn't related to the popular ones is that you can be reliant on them to provide packaged software. Thankfully, this isn't an issue with Deepin because it uses *apt*, the package management system used by Debian and Ubuntu. That means you can install from the vast range of packages provided by them once you've added their repositories. Deepin provides its own graphical package manager. It fits in with the general Deepin appearance and looks functional, but there's no way to add extra repositories – you'd have to do this in a terminal.

Respect the fine manual

Surprisingly, there's no desktop or taskbar icon, or even a menu entry, to open the Deepin manual. Instead, you have to use the menu search to find it, which means you have to know it exists. This is surprising because there has clearly been a lot of effort put into the manual. It covers usage of the desktop as well as the various Deepin applications in great detail. Indeed, it's one of the most comprehensive distro manuals (unless you count the Gentoo handbook) that we've seen.

It's worth mentioning that the Deepin desktop is available separately for installation on other distributions. This may be a good option for those already comfortable with their existing distro, but wanting to try a different desktop. However, you're unlikely to get the tightly integrated experience that you get with the Deepin distro. LXF

VERDICT

DEVELOPER: UnionTech Software Technology
WEB: https://deepin.org/en
LICENCE: EULA for Deepin OS

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

A very slick and easy-to-use distro that's well worth considering. Great manual, too (once you find it).

» Rating 8/10

American Truck Simulator

Management is firm: no new pay rises. So they're terrified that **Andy Kelly** is planning to retrain as a highly paid truck driver.

SPECS

Minimum
OS: Ubuntu 14.04 64-bit, Steam 2.0
CPU: Dual-core 2.4GHz, 64-bit
Memory: 4GB
GPU: GeForce GTS 450 (Intel HD 4000) or equivalent

Recommended
CPU: Quad-core 3.0GHz
Memory: 6GB
GPU: GeForce GTX 760 (2GB) or equivalent

We're driving down a long, remote desert road in the dead of night. There's no other traffic and we can't see anything except the glow of the headlights and the cracked, dusty asphalt ahead. Our attention drifts. *In the Air Tonight* by Phil Collins plays on the radio. Suddenly, something appears directly in front of us. A shape on the road.

We panic, heart racing. Is it a car? An animal? We slam on the brakes and skid to a lurching halt. Then we see it. A tumbleweed rolling lazily across the road in front of us, lit up by the glow of the lights. Laughing at ourselves and continuing towards Los Angeles. There's 30 tons of fertiliser to deliver, and time's running out.

The Boris Brexit Bus pulls up to refill to find the pumps dry and the shelves empty.

This is what passes for an anecdote in *American Truck Simulator*, a game so slow and uneventful that the sudden appearance of tumbleweed is genuinely thrilling. These little moments – military jets streaking across the sky, strange sculptures by the side of the road, trains rumbling past – feel almost like rewards. A brief glimmer of excitement in a long drive across vast swathes of largely empty nothingness.

Like its predecessor, *Euro Truck Simulator 2*, it's a game that's compelling despite its mundane subject matter. It's a fundamentally good game, with weighty, nuanced handling, a deep simulation, and higher production values than most sims. This results in something both very playable and oddly hypnotic. Driving from place to place, obeying the traffic laws, watching the scenery roll by, listening to the radio... it's incredibly relaxing.

There are two main ways to play. One is working as a freelance gun for hire, taking on delivery jobs where your truck and fuel expenses are provided by your employer. This is the easiest, most accessible way to play, and as you level up you can spend XP to unlock more lucrative jobs including longer hauls and fragile or dangerous cargo. These earn you money that can be spent unlocking the other side of the game: running your own business.

Once you earn enough money to buy your own truck – or take out a bank loan if you're impatient – you can start your own company. You choose which city you're based in and can customise and upgrade your truck. Eventually you hire drivers (*Gis a job – Ed*) and create your very own haulage network. It's a fairly involved business management sim, but entirely optional. There's

Customise your rig with a subtle paint job.

Be sure to keep an eye on the road, as well as the sat-nav.

satisfaction in owning a truck and being your own boss, but we prefer being a contractor so don't have to worry about buying fuel or crashing a truck and spiralling into debt when the repair bills come in.

At launch, *American Truck Simulator* came with two states: California and Nevada. More have been added, but it was slightly disappointing that at the time you couldn't drive from coast to coast – officially you still can't even with all the added DLC, though a fan mod is available that enables you to do so (better set aside three hours).

Still, it's a huge space, and you can visit cities including San Francisco, Los Angeles, and Las Vegas – and a number of small towns between. Set on the sun-scorched west coast, the scenery is mostly dusty and desert-like, but they've captured the look and feel of both states nicely. We prefer the overcast, rain-soaked motorways of Europe, but the new setting is detailed and well-made.

Keep on truckin'

Anyone who played *Euro Truck Simulator 2* may find the game a little too familiar in places. The interface and structure are pretty much identical, and it feels like they've picked up the old game and dropped it into the new location. Even different colours or visual flourishes on the interface would have been welcome, just to remind you that it is indeed a new product.

As a result, if you were already burned out on *ETS*, the new setting might not be enough to reignite your passion. It's a very similar experience overall. The trucks – of which there were only two available at launch, and have now been expanded via DLC to seven – don't really feel that different to their European counterparts. It's more like an iteration than a fully-blown sequel.

But if SCS is as tireless and passionate with ongoing updates as it is with *Euro Truck Simulator 2*, *ATS* will only get better as it grows. America is a vast, varied country with a lot of different scenery, and more states to drive through will make for more interesting journeys. For now, this is a polished, strangely enjoyable simulator that you can easily lose hours to. However, if you want to travel further afield, you might want to wait until there are more places to go to. But hold on a minute...

One of Vegas' less glamorous hotels.

Ah, America... land of the oil refineries.

Originally released in 2016, *American Truck Simulator* has enjoyed continuous updates and released over the past five-plus years – something that continues today. This review was originally based on that first release and so it's highly pleasing to report that *American Truck Simulator* alongside *Euro Truck Simulator* are seemingly endless joys to own thanks to the rich content updates regularly made available.

The game came with California and Nevada maps, but additional areas were released as a DLC, including Arizona, New Mexico, Oregon, Washington, Utah, Idaho, Colorado and more recently Wyoming, with Texas to come. Alongside these are more trucks, more haulage types, more tyres and more steering wheels. At time of writing there are large discounts available, too. LXF

Picking up a trailer of valuable cargo (top-end graphic cards, no doubt).

VERDICT

DEVELOPER: SCS Software
WEB: https://americantrucksimulator.com
PRICE: £14.99

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

A solid game on its original release – updates, new maps and more make this an outstanding trucking simulator.

» Rating 9/10

The Perfect Christmas Gift

for just £55.99

plus

FREE! STM MYTH LAPTOP BAG

The STM Myth Laptop Bag is an extraordinarily spacious carrier with cutting-edge contemporary features and a handsome collection of sophisticated tailoring.

The remarkable CableReady system is an incredible tech innovation that sets STM apart from other bags. It enables you to charge on-the-go with lengths of cables out of sight.

The air channel back-panel-and-yoke system is cleverly designed to wrap around you and provide improved weight distribution. In other words: maximum comfort.

Your special Christmas offer

- **Free STM Myth Laptop Bag worth £90**
- **Enjoy 12 months of Linux Format**

  13 magazines for just £55.99 – that's just £4.30 an issue!
- **Receive every issue delivered direct to your door**
- **Get full LXF Archive access with PDFs back to issue 66 (2005)**
- **A thoughtful gift that keeps delivering throughout the year**

Upgrade to include digital for only £10

- **Instant access any time, anywhere**
- **Never miss an issue of Linux Format magazine**
- **Available on iOS or Android**

Order the gift that keeps on delivering

WWW.MAGAZINESDIRECT.COM/LIN/XM21

or call 0330 333 1113 and quote XM21

WE COMPARE TONS OF STUFF SO YOU DON'T HAVE TO!

Roundup

Hugo 0.80.0 » Jekyll 3.9.1 » Pelican 4.6.0 » Staticsite 1.4.1-1 » Mkpage 1.0.0

Mats Tage Axelsson
Mats uses all the tricks he knows, and then some, to keep his old machine going.

Static site generators

Web development is a complex activity, but you can publish a simple website with the help of static site generators and Markdown files.

HOW WE TESTED...

We wrote our content using Markdown, creating five sites. The content was all placed in a local directory, though the files are all synchronised to Git repositories. Static site generators are actually fairly simple collections of scripts that take your text and convert it to HTML using templates and your settings.

Simple systems have a tendency to become big, so we also choose some smaller systems, MkPage and staticsite, for those of you who want to tinker.

We installed the code locally using a package manager where possible and made the structure that constitutes a site on our local machine. Using this as a setup, we created a site for each language and wrote posts and changed themes to our heart's content.

We have used several themes to see what difference it makes and what the challenges are with the different ones. Some themes are Git repositories so this was also tested, as well as copying and adding some code.

There are a ton of ways to make webpages, and most require some sort of build system on the page itself. It also requires a team of web developers for all the features that will drive sales and hook people into your site. If, on the other hand, you decide that some of your creations just need a public space, then not that much is needed.

If you create a site using the big web hosting companies, a text editor is not well supported. To publish these types of sites, you must use their built-in site builders and edit on the page. This makes it really easy to create a good-looking site. Moving to another provider is another issue and if you want to edit locally before you publish, you must cut and paste your final text.

This is where command line lovers can feel rather smug. Using a text editor and Markdown, you can create the entire site. If you want to make it look nice with a theme, you simply need to write some lines in a configuration file and off you go. You have it all at your command line, welcome to the world of static site generators.

Installation

You generally install using a package manager, but in some cases you cut and paste.

Packages to create static sites usually depend on many other packages, as most software does. In the case of the ones in this comparison, there are surprisingly small dependency trees.

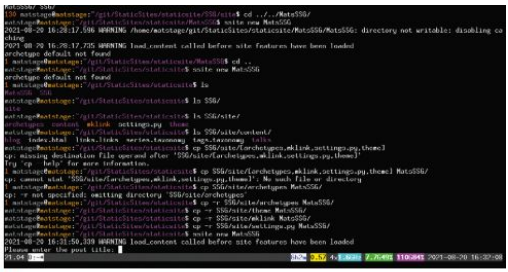
For example, although Hugo is designed in Go, using it does not require the Golang development environment. The same goes for Pelican and Jekyll – though Pelican needs the runtime that is usually on your system.

MkPage depends on Pandoc, it is in fact a front-end of sorts to Pandoc. Staticsite uses several Python libraries to perform its tasks. These two have a very simple install; for MkPage you can just copy the binaries and add them to your PATH variable. Staticsite comes as an *apt* package for Ubuntu users.

Hugo, Pelican and Jekyll have *apt* packages available. Apart from installing using your package manager, Jekyll also has extensions and even themes available this way. For all the other packages, you must install extensions and themes only from the web. Pelican has a big GitHub repository that you can use.

As for MkPage, it uses the Go template engine, like Hugo. For this reason, it should not be an immense challenge to use Hugo themes for MkPage. To install this package, you unpack a ZIP file and add it somewhere in your path. It also comes as a Go *get* package for you Golang enthusiasts (*Mihalis!–Ed*) out there.

When installing from the source, you can find these packages on different Git repositories and GitHub has most of them. Starting with MkPage, you run a regular make sequence to install the package. It's Go-based, so make sure you have those libraries.



Installing staticsite is all good and well, but you may need to copy some files and directories to make a site build. It is more complicated than the other software.

There are also two source trees, the second one for resources where you can hold themes and templates.

As with MkPage, Hugo needs the Golang development environment. Experienced users will use the same method as with any other Go language projects.

Jekyll needs a Ruby dev environment, including the gem binary and the *GCC*, *Make* duo. Run it as you would any other Ruby development project. Finally, Pelican needs Python and a host of HTML libraries, the main one being Jinja2.

The staticsite project is completely Python-based, which makes it simple to use and change if you already have Python set up on your machine.

VERDICT

JEKYLL	**8/10**	**MKPAGE**	**6/10**
HUGO	**8/10**	**STATICSITE**	**5/10**
PELICAN	**7/10**		

The top three are on par when it comes to installing. The only difference is that Jekyll has some in the repo.

Configure your project files

To make a great webpage, you need the project files to be well structured.

The typical website needs many files arranged in a very strict structure to handle styling and to link between pages. Since you are writing the pages yourself, you must have this correct. In most cases, the generators presented here create the entire structure for you. Do remember that the structure will differ slightly between your source files and the finished site.

In the top three, this is the standard way of creating a site. To add new content you can use the subcommands and/or just add the files into the structure and refresh the list. To get started, use all defaults and explore the settings files. You will have an entire site with content when you use Jekyll.

While the procedure varies a little bit, you create a new site with a new command and go from there. In this case, Jekyll is the most complete which also leads to slightly more clean-up.

As alluded to earlier, you can copy a theme into the designated directory and change a pointer in the main config file. This is not always the end of the story. You may run into a case where nothing works; start with the theme to make sure all your values can be evaluated by the theme! You have the option to create a theme yourself. In fact, with both Hugo and Jekyll you can create a structure for a theme with a single sub-command. In Pelican, you are better off using an existing theme and modifying it.

In Pelican, when you start a new site you answer a long line of questions to set it up. This creates the settings that you need to create your initial site.

VERDICT

JEKYLL	**9/10**	**MKPAGE**	**6/10**
HUGO	**8/10**	**STATICSITE**	**5/10**
PELICAN	**8/10**		

The only ones lacking in this area are the MkPage and staticsite packages, which is expected as they are small and new.

Usability

Using generators helps set the style.

Using a static site generator usually involves text files with very little formatting. For this to work, you need to have your content in a format that the system can use. As you have seen earlier, most systems will accept Markdown as the format for your source text.

For the configuration and such, you will need to use what others have written and adjust it. In these files, the format is YAML, JSON or TOML. Most readers should know these formats since we use them extensively for many things. With that said, the formats are easy to learn.

Themes are almost only HTML following the Ninja2 method of adding variables in the document. If you find a theme for Hugo, converting it to Jekyll will probably be a short afternoon practice. There are differences, but you quickly find them if you know the first one very well.

To get started, use one for a while and switch when you have additional needs.

Hugo 9/10

Hugo is extremely simple to get started with. In most cases you create the content with a Hugo command. You then edit that file and try it out. If you run the included local server, you can have the site update live – you won't have time to blink before it is done.

This very popular set of tools is Go-based, but you do not need to know Go to use it. Even the Go templating language is very similar to Jinja2 and other languages. On top of that the programs are binary, so you do not need to have a development environment for Golang on your system.

Setting up your first site is a single command that creates a sample page and the file structure for your site. In the settings file you will find samples of settings for heading, social media links and more.

Jekyll 8/10

Programmers should know that Jekyll is Ruby-based, though ordinary users will not need to know this. What is interesting is that Jekyll is well supported on GitHub, making it a compelling choice if you are hosting on GitHub Pages.

With that said, all the generators create a directory where an entire site sits. In Jekyll's case, that is **_site** by default. The simple way to start a new site creates the site even when you have no content yourself. Since the process creates a configuration file, you need to edit that file to match your needs.

Once you have created the site and fixed everything, you can view your changes locally, live. This makes your creation work simple – just have your browser open on the local site and port and see what happens when you make changes. Templates and styles works as expected and like the others.

Programming language

Your choice may be dictated by the language.

These projects are built in many languages. However, they all end up with the languages needed for your site: HTML, CSS and JavaScript. To get things to work, you need to use web-based techniques. To use the projects, though, you do not need to be fluent in the languages designers wrote them in. Only when you feel you want to contribute does this become a serious consideration.

It does affect contributors, though. You should consider whether you need to tweak the programs themselves or not.

Most work you will do is tweaking aspects of the final result. To do this, you need to know the templating language, which is built on the idea that you can write HTML with variables and conditional statements. The other things you need are YAML, TOML or JSON; most people can learn theses in ten minutes as they are not much more than a data formats.

Commercial support is more important for adoption rates which in turn makes development efforts more intense. This makes language an important aspect to consider. Will the choice of language affect the viability of the project? Do you need to have a lot of developments or do you aim to build your scripts around the generator you choose?

After you have learned the templating language, you can use a shell script to set up the rest of your projects. Obviously, you may want to use the language of the project you are using – Go for Hugo, Python for Pelican and so on. However, you can usually pick whatever language you want to call the binaries and build your site. Have all that in mind when you choose the system for your next web project.

VERDICT

STATICSITE	**9/10**	**HUGO**	**6/10**
MKPAGE	**8/10**	**PELICAN**	**6/10**
JEKYLL	**7/10**		

Do not choose a generator according to your favourite programming language, but consider the needs of your page and your plans.

Pelican 7/10

Since Pelican is Python-based, you can choose to install it with your package manager, with Python, or even use a virtual environment as with other Python projects. You can use Markdown or reStructured text as your input to the site you are making. Themes are Jinja2-based so can be easily adopted from others. The format is nothing new once you have started with any of these packages.

Using the built-in importer, you can import content from blogs including WordPress, Dotclear and RSS-feeds. If you are transitioning, this is a very handy tool to get you started. There are many plug-ins that add web functionality to your site. They also add readers, and if you are handy, you can create your own to make it even easier.

Quickstart creates a directory with a Makefile and conf files, useful for automating your workflow process.

Staticsite 6/10

Labelling itself as simple is true in the sense that it does everything simply. It also means that you must do everything yourself. While you can get a simple site up and running quickly, many useful features are not available.

The software is based on Markdown and Jinja2 and sports a live-preview server. Themes are following the style of Hugo, though directly copying is hardly possible.

With a small community of contributors, you can tell that it is not fully-featured – but if you like to program, especially in Python, this is a great opportunity to learn while building your site and the tools.

Since Staticsite is very simplistic, it requires some more experience. Using the manual, you can set a few simple scripts to have it create what you want. For anyone who likes to set things their own way, here is your chance.

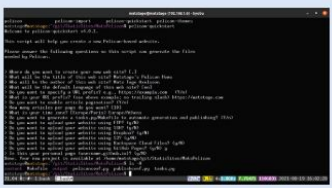
MkPage 5/10

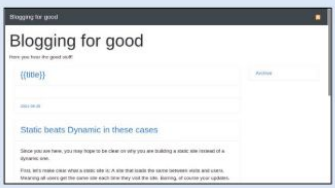
MkPage is made as a collection of tools for the user to make the best of. It uses Pandoc as the backend.

MkPage and its tools look like they are the components of the other tools in this article. This is a deliberate choice by the developers. They call it a deconstructed content system. Despite this, it is actually fairly easy to use, thanks to the use of standards. The templating system is from Go, and supports plain text, Markdown and JSON as input.

You use Mkpage to assemble it all for a webpage. Other components are blogit, mkrss and mkslides. Yes, there is a special component for slides, which again reflects on the Pandoc design.

To have a live preview of your site, you use ws (a Go library). Another component is sitemapper which generates a site map. MkPage is all about the components.





Frameworks

In some cases, you can use web frameworks with these static generators.

There are plenty of themes for the three major ones in this selection. You can see from the selection on their listings that they have embraced known technologies. The most likely choice is Bootstrap, which dominates in the space. If you want to use other frameworks, there are many but you will need to put in a lot of work making your own theme that uses it.

This may be less of an issue than it seems at first. The format for a theme is very similar to any other. You can use this to transfer a theme that you like from one to the other. It is more complex than regular use, but far from a serious programming or web design challenge.

The themes are all supported by a web framework – the earliest of these were Jinja, now Jinja2. Looking through the way it works, there are only details in implementation that differ between them. Any decent computer-savvy person can figure it out rather quickly, all thanks to the adherence to standards.

The existing themes for Hugo, Jekyll and Pelican have many that are based on Bootstrap. You can usually find themes using the many other frontend frameworks if you really want to.

MkPage and staticsite do not have enough supporters to have an impressive collection of themes. You can still use Bootstrap and alternatives, but again it takes some work. If you are a creative person who wants to spend your time tweaking the look, please do. It will benefit everyone and you will get street cred!

Do remember that this is an undertaking – it's not something you can whip up in an afternoon. With that in mind, make your path clear and use ready themes to finish a website and create new themes otherwise.

VERDICT

JEKYLL	**10/10**	**MKPAGE**	**3/10**
HUGO	**9/10**	**STATICSITE**	**3/10**
PELICAN	**9/10**		

The whole reason for these tools is to make the look a separate issue from the content, use themes for this!

Platform support

Things are always easier with a bit of support.

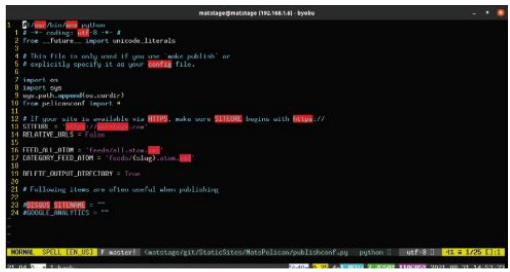
Jekyll is the standard for GitHub, so this makes it a logical choice for many. With that said, your results should be possible to serve with any webserver. This is, after all, the main reason for these systems to exist. You will also drop the need for a database, simplifying your deployment even more.

All webpages your tools generate should only be HTML, CSS and some JavaScript. Thanks to this, any web-hosting company can take your files and display them. You can serve static websites on most big cloud providers, usually on a free plan. The famous ones Amazon, Google and GitHub Pages are examples.

On many hosting platforms, you have to pay higher fees for dynamic content, so mastering this technique to write a website can also save you money.

Most regular web-hosting companies provide one-click installation of WordPress and other advanced functions, but you can get a website up and running completely without it. Deploying your site becomes important when you decide to use the website and change it regularly. Jekyll has a gem: Ruby, this connects to any rack server including Amazon, Heroku and Slicehost.

Hugo has similar features. In both cases, you install the CLI for the service and add the target in the configuration file. With Pelican, you can push the **pelican** directory there and let Heroku take care of the deployment.



Using Pelican, you adjust your settings in publish.conf and use the make command to deliver to site. If you cannot find scripts to deploy to your service, use rsync.

As you can see, choosing the correct generator includes checking which cloud provider you want to use. Picking the correct one will make the process painless and save you a significant amount of time.

If you're a do-it-yourself hero, you can otherwise use *ssh* and in many cases *rsync* to transfer files. Once you have set that up you should never have problems again with deploying your code.

VERDICT

JEKYLL	**8/10**	**STATICSITE**	**5/10**
HUGO	**7/10**	**MKPAGE**	**5/10**
PELICAN	**6/10**		

Choose wisely what provider you are going to use – your choice will have a serious influence on your sanity.

Updating sites

Easy updates are vital.

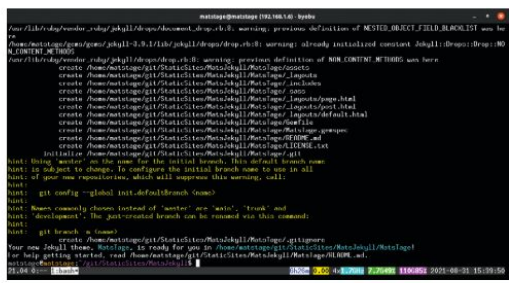
Just adding a post or a page usually means adding the correct file in the correct place and running one command. New themes will make things a little more complex. You will have to check that it works; some themes will need other values filled in to create list pages and the main page.

When you are blogging, Hugo updates according to added files in the correct directory. You actually have the option of adding files or using the *add* command to add a new post. Except for being simple to do, when your site becomes big, you only want to update new content. In the case of Hugo and Jekyll, the default is to cache the content and do version updates.

Adding content from your old WordPress blog is also an issue that you may have considered. Hugo, Jekyll and Pelican have importers that can take your old content directly from a site or a backup. In all cases, there are slight differences between the systems that you may have to handle manually. One example is that each post can only have one category in Pelican, unless you use a plug-in.

If you want to switch your theme, in theory you add all the theme files and change a setting in your configuration file. In reality, you must read through the theme documentation to make sure all the files are included. Links on the main page have to be set by the theme; this is usually the default, but not always.

Uploading depends on the platform, as we said earlier. They all have scripts available to update most platforms. You have several options, including command-line tools from each platform and volunteer scripts available on their respective home pages.



The resulting files you can easily control with git when you create or change a Jekyll site. Even though you have this option for all, there are other options available for you.

Jekyll has an automated script using the continuous integration (CI) of platforms like GitHub, Travis and Buddy. Using this service, you can just `git push` your directories and let the Jekyll magic happen on the service. As we saw earlier, Hugo has a straightforward approach to use the command line from the services.

Finally, upgrading the software is a simple case of using the package manager of choice.

VERDICT

JEKYLL	**8/10**	**MKPAGE**	**3/10**
HUGO	**7/10**	**STATICSITE**	**3/10**
PELICAN	**6/10**		

Updating your site is available for all except MkPage and staticsite.

The Verdict

Static site generators

You can see that these generators have many capabilities despite being minimalistic. Don't get caught up in learning them all at once – you'll never finish your site that way! Migrating from another platform is already built-in, so if you are leaving a commercial offering, check for importers. Hugo, Jekyll and Pelican already have ready scripts for importing from the major providers. Also, keep in mind that they have importers between each other, making it a standard procedure to switch.

All these program suites have a modular design, so when you look through what they do, this becomes apparent and the reason why even more so. They all started using existing code to convert between document formats. Since one output format would be HTML, it just made sense to build an entire site with the code. These modules are what makes importing old content to your new site easy.

Reading formats other than Markdown is another part that is modular. The MkPage suite uses Pandoc, which covers hundreds of formats for both source and target. The big three support two or more other formats than Markdown, plus the formats that their modules support. For *Emacs* enthusiasts, you can use Org-Mode as a source in Pelican, Hugo and Jekyll. You even have major modes for Jekyll and Hugo. With these, you can manage your site from within *Emacs*.

In this collection, the big three are so polished, you can get your site up and running very quickly. The same goes for MkPage and Staticsite, but you will need more skills on the command line. The biggest challenge is to understand how themes work and how to use that to your advantage. With the big ones, you mostly use what others have created. With the new, smaller ones, you need to put in more legwork.

MkPage is small and great for learning how things work, but getting into it will take some time. There are scripts ready to use that can help you understand how the values work with your creation. This is a great way to keep it under your own thinking and control, even when you switch over to other systems. They use the same underlying principles. In fact, the Jinja2 template engine is so commonly used that you will know half of it when you have mastered it.

1st Jekyll 9/10

Web jekyllrb.com **Licence** MIT License
Version v3.9.1
The fact that Jekyll powers GitHub Pages may make up your mind.

2nd Hugo 8/10

Web https://gohugo.io **Licence** Apache-2.0
Version v0.80.0/extended
Hugo has so much choice in content types that it may feel overwhelming.

3rd Pelican 7/10

Web https://blog.getpelican.com **Licence** AGPL-3.0
Version 4.6.0 Thanks to using Python, Pelican has many plug-ins that enhance its functionality.

4th MkPage 5/10

Web https://caltechlibrary.github.io/mkpage
Licence Copyright 2021, Caltech **Version** 1.0.0
This software shows very little consideration to first-time users' needs.

5th Staticsite 5/10

Web https://github.com/spanezz/staticsite
Licence GPL-3.0 **Version** 1.4.1-1
Though thin on features, Staticsite will take you directly to your editing.

» ALSO CONSIDER...

Gatsby (**https://www.gatsbyjs.com**) creates static sites with the same techniques and ideas as the rest of this collection of scripts. The big difference is that Gatsby uses React heavily.

This makes it possible to create store pages, WordPress and other content-driven websites. BUt it also makes it necessary for you to know React. Nodejs must also be installed. With all these demands, you are moving up to a much higher style of web development.

If you wanted to use a static site generator to make things simpler for you, you've come full circle and you are starting to program a webpage. Even though this package creates a static page, this is going a step further in terms of learning. Most people will be happy to stay with one of the other options for at least a while before needing to move to this level of power. It is very powerful for developing webpages, but overkill for simple content.

LINUX ❤ WINDOWS

Old rivalries have been forgotten and ancient boundaries blurred. **Jonni Bidwell** investigates this new Redmond-Penguin harmony.

Back in 2015 Microsoft decided that it loved Linux. It loved it so much that it built a whole Windows Subsystem for Linux (WSL), which enabled Linux programs and development stacks to run natively. The official statement even involved a heart emoji, which we would have reproduced here but putting in-line images in the body text apparently causes alarm bells and wisps of smoke at the printers.

Anyway, heart or no, some people were sceptical of Microsoft's intentions, with memories of the "Embrace, Extend, Extinguish" mantra and the 1995 Halloween papers still fresh in their minds. But it does really seem like Microsoft wants to accommodate Linux users (well, developers mostly), rather than force a mass defection.

A successor, WSL 2.0, was announced in 2019, which was built around a real Linux Kernel, rather than a *Wine*-like (or whatever is the inverse of *Wine*-like) translation layer. So WSL 2.0 brought faster performance, swifter filesystems and increased application compatibility. Back in April this year, an exciting new feature was announced. WSLg, enables graphical tools to run seamlessly on WSL. No need to shoehorn an X server on Windows, no need to redirect *PulseAudio* – heck, it even works with Wayland. So not only can you run Bash on Ubuntu on Windows (WSL's working title), you can also run *Blender*, *GIMP* and *Krita*. We'll see how easy it is to set this up on Windows, how it's a great way to learn Linux, and how to do some weird and wonderful stuff with it.

Finally, we'll look at some old and new efforts Linux has made to bridge the divide with its proprietary counterpart. Filesystems, networking, bootloaders. Yes, we've broken them all at some stage, but now things are mature and established enough that everything should just work with minor configuration changes.

Platforms, Trusted Platforms and Windows 11

Windows hardware requirements have always promoted confusion, sand this time is no different...

When Windows 8 was announced, there was a concern that Microsoft would use Secure Boot, an optional feature of the then new UEFI (Universal Extensible Firmware Interface), to restrict the installation of other operating systems. Secure Boot only allows booting EFI images that have been signed by a key enrolled in the UEFI. Since almost all hardware ships with a public signing key from Microsoft, it's easy to see where this concern came from. However, to help Linux distros (or any software that needed its own bootloader) deal with Secure Boot, Microsoft used its magic key to sign a small program (they probably wouldn't sign something big and complicated) called *Shim*.

Shim is a first-stage bootloader used by many distros (including Debian and Ubuntu) to launch their own (second stage) bootloaders. Secure Boot works by each stage checking the signature of the following one before executing it, thus establishing a root of trust back to the original signing key. So distributions can embed their own keys into *Shim*, and have this distro-specific *Shim* package signed by Microsoft. *Shim* can then check the GRUB EFI image, signed by the distro key, and boot can proceed. It's possible to sign kernel images too, as well as subsequently loaded modules and firmware, so if you trust cryptography and the Microsoft Signing Authority, then Secure Boot makes it really hard for malware to be loaded anywhere in the early boot process.

A question of trust

Of course, not everyone's going to trust Microsoft, or indeed Secure Boot. And that's okay, because in 2013 when Microsoft stipulated that "Windows 8 Ready" hardware should ship with Secure Boot enabled, it also made the provision that it should be possible to disable it, and also that it should be possible for users to install their own MOKs (Machine Owner Keys) and use them to sign whatever bootloader they wanted, so that ultimate trust (but also ultimate responsibility) lay with the user.

"Windows 10 Ready" hardware was subject to similar conditions, except the requirement that Secure Boot be mutable was reduced to a suggestion. Still, we've never come across a single Windows 10 machine where Secure Boot could not be disabled. And if you build your own systems, then you have nothing to worry about. The "Ready" conditions are for OEMs that ship systems with the OS already installed. Now with Windows 11 upon us, it's no longer Secure Boot clauses that are being scrutinised , but those relating to TPM chips.

Tiled start menus and the Windows desktop may bring you no joy, but look – Ubuntu can cheer you up.

Trusted Platform Module (TPM) chips are tiny processors that have a small privileged memory store, which applications can use to store keys, authentication data or anything that they don't want other applications nosing at or tampering with. Besides some (fairly strict) minimum hardware specifications, for a PC to officially support Windows 11 it must also support TPM 2.0. At the moment, it's still possible to install Windows 11 via the official ISO if your device doesn't meet the TPM (or other) requirements, but Microsoft says such installations are unsupported, and may not be privy to security updates further down the line. Indeed, at the time of writing we've seen Insider builds of Windows 11 running on a Pi 400, as well as a Nokia Lumia 950XL (one of the last devices to run Windows Phone).

GIVE ME ALL YOUR TPM

TPM chips have been around for most of the past decade. TPM 2.0 was introduced in 2014, and most motherboards from 2016 onwards include one. TPM can be implemented in firmware too (so-called fTPM), at a slight cost to security, since some new Spectre/Meltdown-type attack could, in "theory", be leveraged against it. Still, fTPM is good enough for Windows 11's requirements. You might need to enable TPM via the UEFI (classic BIOS is also not supported), where it goes by so many names that Microsoft made a friendly help page (see **https://bit.ly/lxf282-mshelp-tpm2**)

TPM is fully supported on Linux, and can be used to secure SSH keys (see **http://blog.habets.se/2013/11/TPM-chip-protecting-SSH-keys---properly**), unlock LUKS encrypted volumes (via systemd-cryptenroll or Clevis) or even make Secure Boot even securer (**https://threat.tevora.com/secure-boot-tpm-2**). There are separate software stacks for TPM 1.2 (TSS aka TrouSerS) and TPM 2.0 (*tpm2-tools*) and there's a nice summary of both on the Arch Wiki page at **https://wiki.archlinux.org/title/Trusted_Platform_Module**.

Linux in Windows

Never mind Windows 11, the latest version of the Windows Subsystem for Linux is where it's at.

Are you part of the slightly sinister sounding Windows Insiders Program? Have you installed an even more sinister-sounding Preview Build of Windows 10 (at least build 20262)? Then it's easy to run Linux as part of WSL 2.0: just open an administrator-privilege Windows command prompt and run `wsl --install`. This also works on Preview Builds of Windows 11. A few clicks and pops and a restart later and you'll be in business.

Actually the GUI App Support download is pretty large, so now might be a good time to make a cup of tea. On return WSL, Microsoft's Virtual Machine Platform, their custom Linux Kernel and Ubuntu should all have been downloaded. Other distros are available and one can specify, for example, `wsl --install -d Fedora`, to install Fedora instead, or additionally if you already have WSL Ubuntu installed. Users of non Preview Builds can either join the Insiders Program and upgrade to one, or follow the manual installation steps below.

HASSLE-FREE SETUP

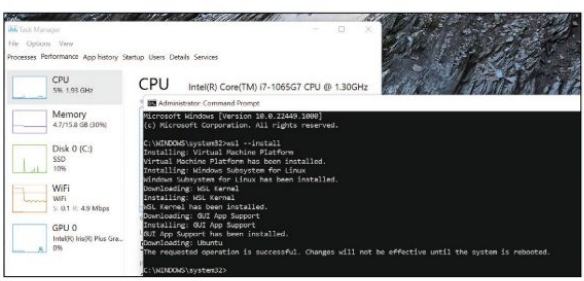
"One of the amazing things about WSL 2.0 is that it supports GUI tools without any extra configuration"

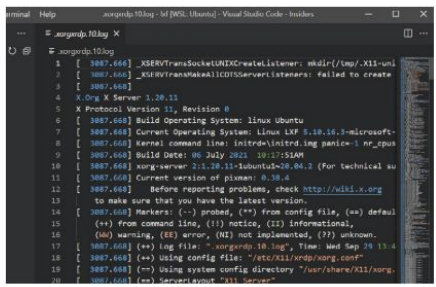
Start by firing up Administrator-powered *PowerShell*, then acknowledging that the command line can get ugly on Windows too, enter the incantation:

```
> dism.exe /online /enable-feature /
featurename:Microsoft-Windows-Subsystem-Linux /
all /norestart
```

This will install WSL 1, at which point you might wish to reboot and play with that. Or you could continue to get the latest incarnation. If you're not running Build 18362 (check by running *winver.exe*) or higher (or 19041 or higher form ARM64) you'll need to persuade the *Windows Update Assistant* to get you there. To enable

It's time to upgrade Windows 11, by adding a Linux kernel and some Microsoft magic.





Now you can pore over logs and try and see what went wrong with RDP.

WSL 2 we first must enable the Virtual Machine Platform, which requires this doozy of a line to be entered at an Administrator PowerShell:

```
> dism.exe /online /enable-feature /featurename:Virtu
alMachinePlatform /all /norestart
```

One restart later and you'll be running the subsystem sequel. To get Microsoft's latest frankenkernel, download and run the package at **https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi**. There's a separate package for ARM devices, so change x64 for arm64 in the previous link if that applies to you. Before we get to installing Linux we need to set WSL 2 as the default version from PowerShell, easy enough:

```
>wsl --set-default-version 2
```

Then get ye to the Microsoft Store (**https://aka.ms/wslstore**) and install a distribution. The recognisable flavours are all free, but if you're feeling flush you might want to try the Microsoft Research's WSL-tailored Pengwin distro, currently on sale for about half of its £16.74 (*or 2.5x a copy of LXF–ED*) price tag.

Create a standalone account

Whatever you choose, and however you got there, the first step post-install is to follow the prompts and set up a username and password. This is entirely unrelated to any other accounts you have set up on Windows, and also separate from WSL accounts set up by other Windows users on the same machine. In other words WSL machines are effectively single-user affairs, since they're the concern of a single Windows user. Then just as if it were a real Linux machine, you should do a system update:

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

Where you go from here is up to you. You could just install *Vim* and the **build-essentials** package and get hacking on your latest C project. Or you might want to install the *Nvidia Container Toolkit*, which sets up everything you need to run GPU accelerated Docker

images, so you can use *CUDA*, *PyTorch* and *TensorFlow* to machine-learn your way to freedom.

Or you might want to try something else. One of the amazing things about WSL 2.0 is that it supports GUI tools without any extra configuration. It even has its own Wayland compositor. In the previous version, graphical tools could be used, but required an X server (such as GWSL, VcXsrv or X410) to be running on the Windows side. A quick poll of *LXF* staff suggests that our favourite GUI tool is *Microsoft Edge*, and our favourite type of humour is sarcasm. So if you wanted to run the Linux edition of Microsoft's browser in a Microsoft-flavoured version of Linux, you could go right ahead and do:

```
$ sudo apt install software-properties-common apt-transport-https wget
$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/edge stable main"
$ sudo apt install microsoft-edge-dev
```

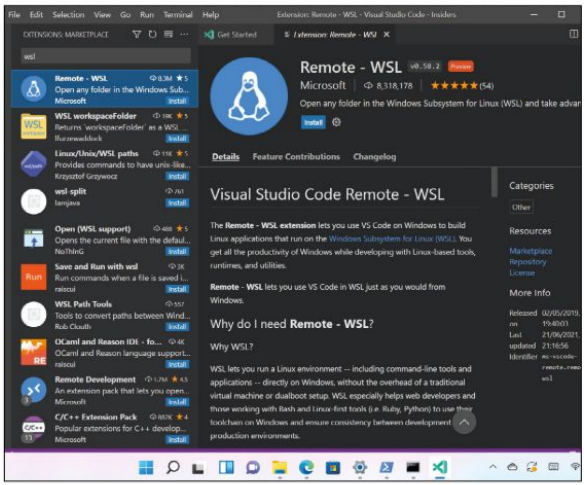
Now *Edge* should appear in the start menu, or you can start it with a simple `microsoft-edge`.

In this way, one could also install such Redmondian wonders as *Microsoft Teams* or *Skype*. Or perhaps something that isn't so Windows-centric. Popular open source programs are just that, so the likes of *GIMP*, *Blender*, *VLC* et al all have official Windows equivalents. If for some reason these don't work well, then now you have options. The Transmission *BitTorrent* client is perhaps an exception – it has a Windows port, but it's described as an early preview. You may have your own Windows torrent client preferences (*uTorrent* is popular), but if not you can enjoy the solid reliability of the Linux version. 'Tis but a matter of:

```
$ sudo apt install transmission
```

Visual Studio Code (or the VSCodium fork) are becoming increasingly popular on Linux. And it's fast becoming the code editor of choice for Raspberry Pi users. We can install the Linux version in much the same way as we installed *Edge* above, but we can also do something different. Recent *VS Code* versions support the WSL-remote extension, so you can install the Windows version (from **https://code.visualstudio.com**) and then have easy access to your WSL files.

For this to work ensure "Add to PATH" is checked in the Select Additional Tasks dialog at the start of the installation. Then from the Extensions marketplace, search for Remote Development and install the extension pack (which enables remoting to WSL, Containers or over SSH). According to the docs at **https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode** you should now be able to start *VS* code from within WSL by opening the Ubuntu command line and running `code .`. This didn't work for us, but we were able to use F1 to bring up the command palette in *VS Code* and search for `Remote-WSL`, which reveals some of its many functions. These include being able to open a folder in WSL, complete with its own Terminal.



Having a whole Marketplace inside a code editor isn't everyone's cup of tea, but you'll find WSL extension fortifying and refreshing.

If you run the `top` command within a freshly started WSL 2.0 instance, you'll see there's not really all that much running. A couple of *init* processes, *Bash* and *Top* itself. Not a hint of the bloat some accuse Microsoft of. This is a slight subterfuge, because a lot of the magic is happening in an invisible service virtual machine. This runs a containerised distro based on Microsoft's own CBL Mariner Linux (developed for its Azure cloud) which handles Wayland, X, PulseAudio and beams it back into Windows via *FreeRDP*. Read more about this marvel, known as WSLg, at **https://github.com/microsoft/wslg**.

» FOREIGN FILESYSTEMS

From the Linux side, your Windows C: drive and everything in it is available at **/mnt/c** and from the Windows side, you can see your Linux filesystem(s) from Explorer by following the Linux shortcut (it has a penguin icon) below OneDrive, This PC, Network and what-have-you. While handy, you shouldn't let Linux programs do I/O heavy operations on **/mnt/c**. Why? Well read on...

If you run the `mount` command, you'll see that **/mnt/c** uses some sort of exotic Plan 9 (9p) filesystem to mount a **drvfs** volume. You'll also see that the root Linux filesystem lives on its own block device (**/dev/sdc** or such) and that there are mount points for WSLg (the helper distro that enables tools to run seamlessly with graphics and audio).

In WSL 1, **drvfs** is used directly, which means that performance between Windows and Linux filesystems was a little more performant. WSL 2's use of a 9p server to wrangle these transfers slows things down, but also enables more flexibility and security. For this reason, WSL 1 will outperform WSL 2 at 'across the boundary' transfers. So if you need to run, say, a database in WSL but for some reason its files must be visible in Windows, then you should downgrade to WSL 1. It's more reasonable to try and keep heavy transfers within the WSL bubble. It's even possible to add additional drives to your machine and format them as Ext4 (or whatever is your favourite Linux filesystem) from within WSL. You won't be able to see them from the Windows side (at least not without additional tooling), but transferring large files or doing lots of random I/O should be much swifter.

Year of the Linux desktop (on Windows)

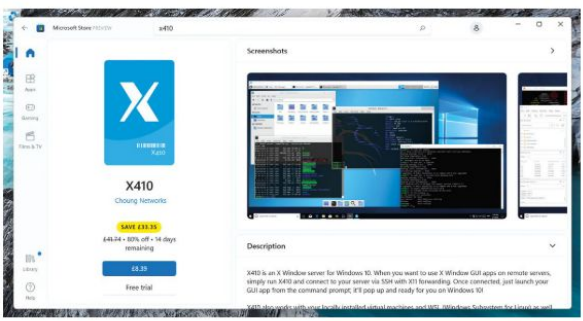
Running graphical applications is impressive, so the next step is to try and run a complete desktop. What are you waiting for?

Getting an actual Linux desktop running on WSL 2.0 on Windows 11 proved a little tricky. In fairness, WSL was built to run applications, not to act like a conventional virtual machine. It's not really surprising that Gnome and KDE won't work. They rely on all sorts of daemons and services running in the background holding all the components together. Brave souls have managed to get the Xfce desktop working nicely on Windows 10 with Ubuntu 18.04, but this didn't work for us on Windows 11 with Ubuntu 20.04. If you're running Windows 10, then you'd be well advised to try out running Xfce via the X410 server. It should be a matter of

```
$ sudo apt install xfce4
```

and then setting the `DISPLAY` environment variable correctly. A major change in WSL 2 is that 127.0.0.1 in WSL doesn't refer to the real loopback interface in Windows (where the X410 server would be listening). So we need to figure out the WSL address and update `DISPLAY` accordingly. Per the instructions at **https://x410.dev/cookbook/wsl/using-x410-with-wsl2** this can be gleaned from **/etc/resolv.conf** on WSL, and then all that's left is to tell X410 to allow "Public" connections. You'll probably want to use *Windows Defender Firewall* to restrict this (if your machine isn't otherwise firewalled). Then run `xfce4-session` within WSL.

X410 is a well-regarded and occasionally discounted X Server for Windows 10. We need an X411, though.



Taking a different approach

While not technically in the remit of WSL, it's still nice to have a Linux desktop on Windows 11 (if nothing else then to show how the new-fangled desktop measures up to something tried and tested). Since using a native X server on Windows didn't work for us, we figured we'd try a different approach: using Windows own Remote Desktop Protocol (RDP) to connect to a desktop running within WSL2. To do this we installed Xfce in Ubuntu using the command above (note that in some situations you may need to install `xfce-terminal` explicitly as well), and then installed and started *Xrdp*:

```
$ sudo apt install xrdp
$ sudo /etc/init.d/xrdp start
```

Note the old Sys-V style init script. WSL doesn't use Systemd, which will surely be good news to some people, but is also part of the reason you can't use Gnome on it right now. Anyway, we're getting sidetracked. So start *Remote Desktop Connection* in Windows and connect to **localhost:3389**. You'll be

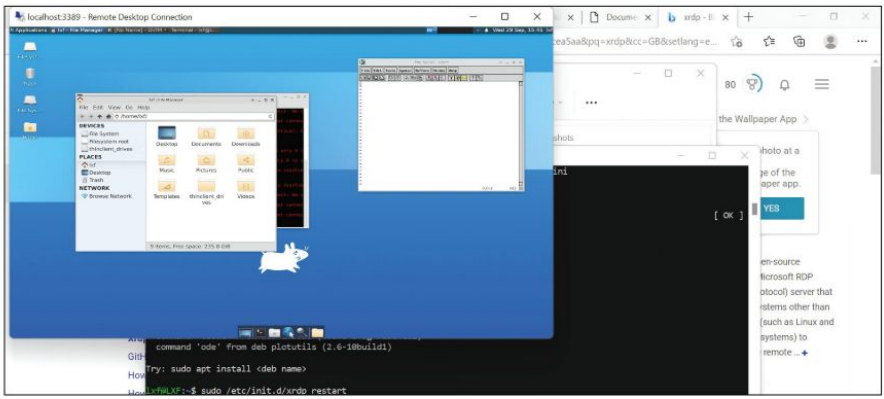
HARDWARE-ACCELERATED WSL

WSL supports running with a virtual GPU, so that Linux programs can take advantage of high-powered graphics cards and run compute jobs (not games at this stage) at near-native speeds. There's also a DirectML backend for *TensorFlow* (see **https://github.com/microsoft/DirectML**), enabling speedy machine learning experiments on Direct3D 12-supported hardware.

Further to this, in spring 2021 the d3d12 driver became an official part of the open source Mesa 21.0 stack. To make use of this, and speed up your *Blender* renders or whatever, you might need to update your Windows GPU driver to one supporting WDDM 3.0.

Nvidia's WSL CUDA drivers are considered public preview, and should be delivered to Windows Insiders through the normal Windows Update channel. Otherwise they're available at **https://developer.nvidia.com/cuda/wsl**. Owners of AMD graphics hardware can get the preview drivers at **www.amd.com/en/support/kb/release-notes/rn-rad-win-wsl-support**. Intel's can be found at **https://bit.ly/lxf282-intel-drivers**.

Since only Ubuntu LTSes are available on WSL, they miss out on all these exciting developments. So you could install Arch, or some other bleeding-edge distro. Or you could avail yourself of Canonical's Ubuntu on Windows Community Preview. Read more and follow the download links at **https://ubuntu.com/blog/announcing-ubuntu-on-windows-community-preview-wsl-2**.



Eventually our efforts paid off and we were greeted with the minimal, yet glorious Xfce desktop.

warned that the identity of the remote computer cannot be verified, but since the remote computer here is in fact local this is nothing to worry about. A login screen should open, from which you should select the Xorg session and use your WSL credentials to log in. At first this didn't work for us, and we got nervous because time was getting short (*I think we're in negative time – Ed*). We encountered a mildly encouraging blank screen that then vanished, dashing our hopes. Then we remembered that X sessions are complicated things, so we had little hope that the following hack would actually work. But it did! And hopefully it does for you, too. Make a trivial session file in your home directory (on WSL) as follows:

```
$ echo xfce4-session > ~/.xsession
```

And now try and log in again. If it doesn't work you can always log out of WSL and shut it down and restart it by running:

```
$ logout
> wsl –shutdown
> ubuntu
```

Our session seemed to die whenever the Windows screensaver activated, necessitating this step on a number of occasions. Once you get it working you'll notice that the image quality isn't quite pixel perfect, but this can be tweaked by giving the server more bandwidth. Exit *Xrdp* and edit the config file with

```
$ sudo nano /etc/xrdp/xrdp.ini
```

Next, find the `max_bpp` value, change it from 32 to 128. Below this add the line

```
xserverbpp=128
```

Now restart *Xrdp* with:

```
$ sudo /etc/init.d/xrdp restart
```

and you should have a slightly sharper, slightly more responsive desktop.

Learning tool

If you're something of a Windows expert, but don't have much familiarity with Linux, then having this remote desktop on hand will be a great learning tool. It's a hip alternative to a virtual machine. And just like a virtual machine you can export the installation and run it on another machine, or on the same one if you want a handy backup before you try some adventurous command line juju. To export a WSL image as a tarball run the following from a Windows *Command Prompt*:

```
> wsl.exe --export Ubuntu c:\ubuntu.tar
```

replacing `Ubuntu` if you used another distro. You can keep track of which WSL distros you have installed with

```
> wsl.exe --list --all
```

Then, if you have cause to re-import a tarball you can give the distro a different name:

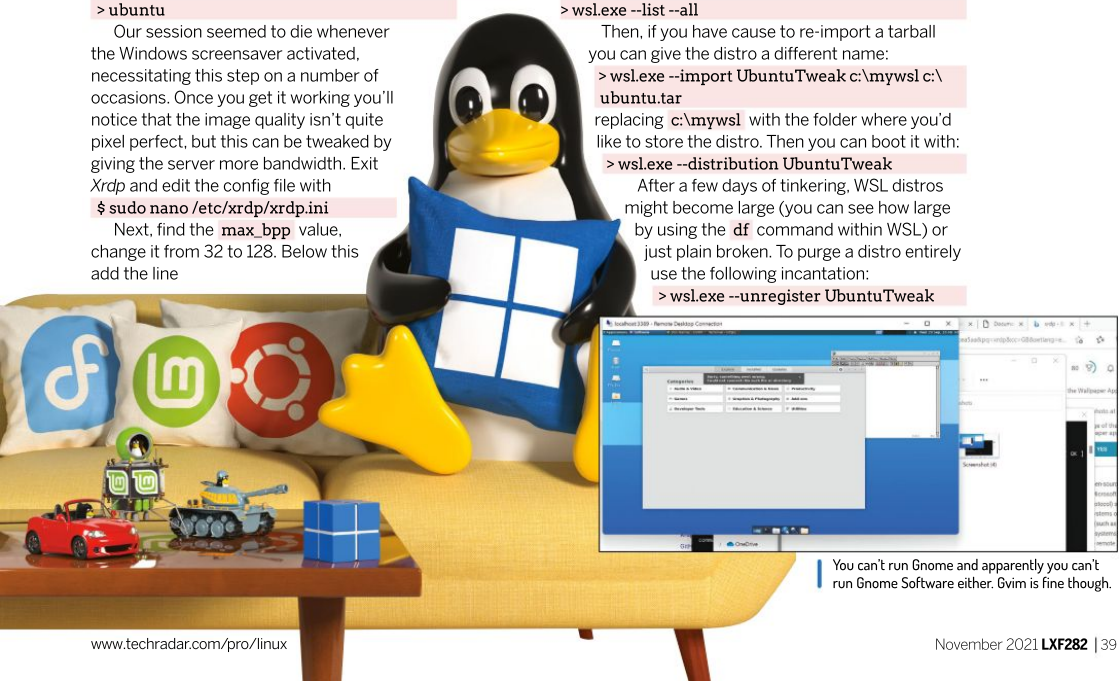
```
> wsl.exe --import UbuntuTweak c:\mywsl c:\
ubuntu.tar
```

replacing `c:\mywsl` with the folder where you'd like to store the distro. Then you can boot it with:

```
> wsl.exe --distribution UbuntuTweak
```

After a few days of tinkering, WSL distros might become large (you can see how large by using the `df` command within WSL) or just plain broken. To purge a distro entirely use the following incantation:

```
> wsl.exe --unregister UbuntuTweak
```



You can't run Gnome and apparently you can't run Gnome Software either. Gvim is fine though.

Duelling OSes

Linux has always tried to co-operate with Windows oddities, and despite common gripes, it does an amazing job.

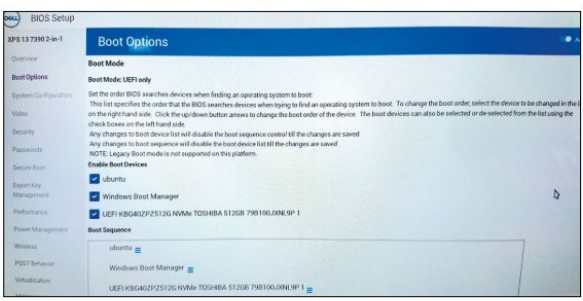
Owners of a modern UEFI system should (as long as you can get into the UEFI settings) be able to revert any unwanted changes caused by installing a new operating system. The UEFI boots an image from the EFI partition or live media, and userspace tools (both on Windows and through *efimgr* on Linux) can change the boot order and in some cases break things. It's generally expected (but not necessarily desired) that a newly installed OS will put itself at the top of the boot ordering. Often this results in Windows booting immediately and the only way to get to the UEFI settings to revert this is to hold the Shift key while shutting down from the start menu. An option to reboot to settings should appear.

Our experience was, mercifully pain-free though. We upgraded Windows 10 on a 10th generation Dell XPS set up to boot Ubuntu, and lo and behold the default boot choice (GRUB) was left intact. Not only did Ubuntu boot correctly, but the old GRUB entry for chainloading Windows also worked. We think we got lucky here, but even if we didn't, booting Windows would just be a matter of entering the UEFI settings or summoning a boot menu at startup. If UEFI didn't give us these options, then we can reboot to the firmware setup with

```
$ systemctl reboot --firmware-setup
```

Modern UEFI makes it difficult for Windows to prevent Linux from booting.

For older BIOS machines the situation is different. Each hard drive can have its own Master Boot Record (MBR), and any OSes installed can put their bootloaders here. Well, the first part of their bootloaders, the MBR is only about 440 bytes long, and modern bootloaders like GRUB are complicated, so the MBR contains a skeleton program that tells the machine where on the hard drive the rest of the bootloader can be found.



Grub version 1 also had a Stage 1.5, which filled about 30KB directly after the MBR and contained the appropriate filesystem driver for the partition containing Stage 2. Grub v2 doesn't need Stage 1.5, as Stage 1 points directly to the logical block address containing Stage 2 (without worrying about filesystems), and then the Stage 2 will load the main config file **/boot/grub/grub.cfg** as well.

We've long recommended keeping separate OSes on separate drives, and one reason for this is to mitigate against one OS's bootloader treading on the toes of another. This is as much a problem between different Linux distros as it is between Linux and Windows. Linux distros are generally good enough to give you a boot menu, but entries for secondary distros won't respect any of their settings in **/etc/default/grub**, which may cause issues. Also, if one of these distros is Arch Linux, it often doesn't get detected by the `grub-install` scripts. Installing the `lsb-release` package on the Arch side should solve this. Naturally Windows has no qualms about obliterating any other bootloaders on its drive. It's easy enough to get GRUB back when this happens, but better to avoid it happening by giving Windows its own drive.

DANCING WITH THE STARS

"Samba is vital for anyone who wants to use a Linux machine on a Windows network"

Rogue Windows updates

But this might not be enough. We've read plenty of posts online saying that Windows updates have boldly gone where they're not supposed to go, and we've been around long enough to have seen this behaviour first hand. Planting recovery partitions on other drives, corrupting LVM and RAID volumes, and all kinds of other unorthodox behaviour. Windows has nothing to gain from this, but living alongside another OS was never in its remit, so in a sense this is to be expected.

Besides battling bootloaders and (maybe) dodgy .docx files, one of the major points of friction between Windows and other OSes is the NTFS filesystem. It's not just a concern for Linux – getting macOS or Android to write to these volumes was always tricky too, at least without proprietary tools. On Linux we at least had the userspace NTFS-3G driver, but this was not the most performant. Mercifully, help is at hand in the form of a new kernel driver (see the box, *right*). Also, we shouldn't bash NTFS-3G. The official WSL documentation warns

that the converse situation, viz. transfers from within WSL to without won't be blazing fast either. This is a slight regression from WSL 1, which thanks to not having a real Linux Kernel was able to streamline transfers across filesystems. Still, certain I/O-heavy workloads to Linux filesystems that were hobbled under WSL 1, should now do much better.

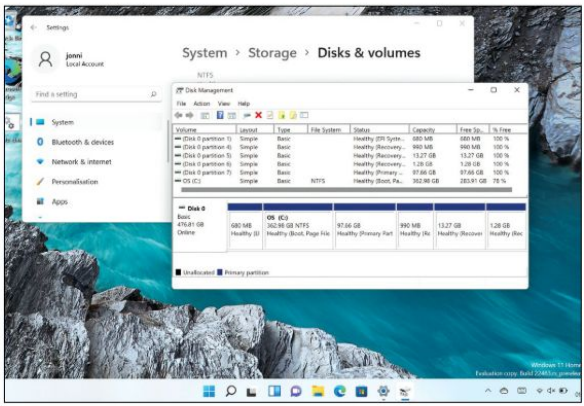
Besides NTFS (*see below*), Paragon Software has an agreement with Microsoft that enables it to market and licence its own exFAT implementation(s). This would have been a boon for Paragon, but Microsoft effectively open sourced exFAT in 2019. More precisely, it made public the filesystem's specifications and offered free licencing to any member of the Open Invention Network (OIN, holders of a defensive patent pool to which Microsoft and others have contributed their IP). The actual work of writing a driver it left to the community, and this, thanks to Samsung, was added to Kernel 5.7. Again, there was previously an exFAT kernel driver which first appeared in version 5.4. This was based on an old driver for one of Samsung's Android tablets, the code accidentally ended up on GitHub in 2013 and was later open sourced by Samsung. That code, described at the time as "horrible" by Greg K-H, ended up in the Kernel's Staging tree where it was improved, before being replaced by Samsung's newer code drop.

Network filesystems have been improved by Samsung too. Kernel 5.15 includes a new driver, KSMBD, for serving shares over the latest version of Windows Server Message Block (SMB3) protocol. This should make for faster transfers in the immediate term, but long term the goal is to have a leaner project that's easy to add new features to as the protocol evolves. This is in contrast to the userspace Samba, a massive (and venerable) project that deals not just with serving files over SMB, but also client tools, authentication, Active Directory and much more. So KSMBD is designed to work in harmony with Samba, rather than replace it. High on KSMBD's list of things to do is to implement RDMA (Remote Direct Memory Access) support, also known as SMB Direct, which enables file transfers to skip protocol overheads, currently a bottleneck for mobile and embedded systems.

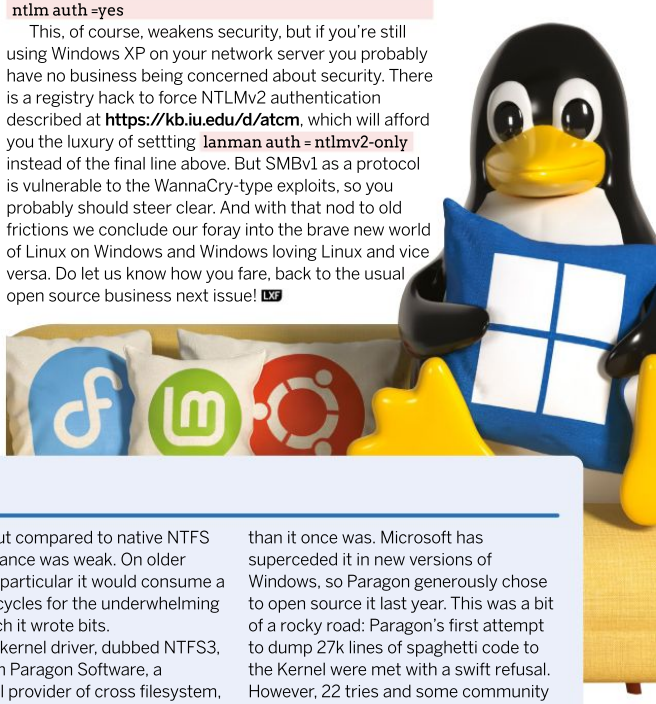
Samba is vital for anyone who wants to use a Linux machine on a Windows network and for the most part it works out of the box. There are some edge cases though. For some reason many of our readers have run into the issue of old 'LANman' password authentication no longer being supported, so out of the box it's not possible to connect to SMB shares on Windows XP (or older) machines. These machines also, by default, use an older version of the protocol, and to make *Samba* play along the generally accepted solution is to add the following lines to **/etc/samba/smb.conf**:

```
server min protocol = NT1
lanman auth = yes
ntlm auth =yes
```

This, of course, weakens security, but if you're still using Windows XP on your network server you probably have no business being concerned about security. There is a registry hack to force NTLMv2 authentication described at **https://kb.iu.edu/d/atcm**, which will afford you the luxury of settting `lanman auth = ntlmv2-only` instead of the final line above. But SMBv1 as a protocol is vulnerable to the WannaCry-type exploits, so you probably should steer clear. And with that nod to old frictions we conclude our foray into the brave new world of Linux on Windows and Windows loving Linux and vice versa. Do let us know how you fare, back to the usual open source business next issue! LXF



Windows Disk Management tool begrudgingly accepts our Linux partition is there, but offers no info.



» PARAGON NTFS3 DRIVER

After years of having to make do with the NTFS-3g FUSE (Filesystem in USErspace) driver to read and write to Windows NTFS volumes in Linux, a driver is finally coming to the kernel. Actually there has for a long time been a kernel driver (since 2001 in fact), but it only ever provided read-only support and was overlooked by most people. NTFS-3g was a valuable crutch for people who worked on dual-boot systems, but compared to native NTFS its performance was weak. On older systems in particular it would consume a lot of CPU cycles for the underwhelming rate at which it wrote bits.

The new kernel driver, dubbed NTFS3, comes from Paragon Software, a commercial provider of cross filesystem, partitioning and network management tools. NTFS has been around since 1993, and is of much less commercial interest than it once was. Microsoft has superceded it in new versions of Windows, so Paragon generously chose to open source it last year. This was a bit of a rocky road: Paragon's first attempt to dump 27k lines of spaghetti code to the Kernel were met with a swift refusal. However, 22 tries and some community advice later, the code has finally landed in Kernel 5.15, which will be released by the time you read this.

Ellora James is a fourth-year student who's taking a university degree course on ethical hacking.

» HELLO WORLD OF RASPBERRY PI

I was first introduced to the Raspberry Pi back in high school, and I've been hooked ever since printing my first "Hello World" in Python. Recently, I started a YouTube channel (**www.youtube.com/c/ElloraJames**), where I share my latest Pi projects.

The accessibility of the Raspberry Pi and the supportive community behind it has no doubt played a major role in my decision to pursue a career in ethical hacking. Had I not borrowed a Raspberry Pi from school and started learning Python, I may never have discovered my love for programming. Ultimately, that led to me choosing to study computer science in school – something that I had never considered before.

Starting my YouTube channel has further shown me just how supportive the community is. After just one video, I was receiving great feedback, and people are always happy to leave helpful tips in the comments. It's also opened up opportunities for me to feature in *The MagPi*, on the Raspberry Pi Foundation's own YouTube channel, and of course here in *Linux Format*!

The main appeal of the Raspberry Pi to me is the creative aspect to it. The possibilities of what you can create with the Raspberry Pi really are endless. All you need is a bit of imagination, creativity and some problem-solving skills. I'm really enjoying creating both the projects and videos for my channel, and I'm looking forward to continuing working on it and seeing what projects I end up doing next!

Astro Pi 2021/22 launches with new kit

Seven super-successful years on and the Astro Pi models get a hefty upgrade for the next competition.

The Astro Pi Challenge is a competition that's been running since 2015 with the European Space Agency Education Office and the Raspberry Pi Foundation. It started as part of UK astronaut Tim Peak's Principia mission, taking two space-hardened Pis with him to run code developed by competing school children. Since then 54,000 students from 26 countries have written code that has run in space.

Seven years on and the old Astro Pi models are getting a major upgrade. After two years of secret development, the upgraded Astro Pi VIS and Astro Pi IR computers are ready to blast off! At their heart is the latest Raspberry Pi 4 Model B with 8GB of memory, the Pi HQ Camera, Google Coral ML accelerator and colour, luminosity and IR sensors. These are in addition to the usual gyroscope, humidity, accelerometer, magnetometer, temperature and pressure sensors.

The new models are waiting to run Mission Space Lab submissions made by 20 October 2021 and Mission Zero entries by 18 March 2022. Find out more about how to enter here: **https://astro-pi.org**.

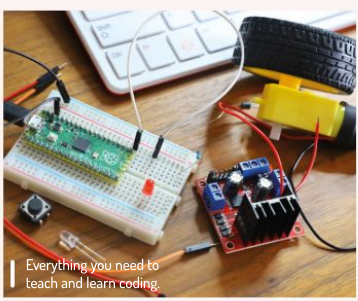
Superpowered and super-specced, the new Rapsberry Pis are in orbit now and waiting to run your code...

CREDIT: Raspberry Pi Foundation

EduPi USA

Stateside learning.

The Pi Foundation is constantly looking to expand its educational reach and now in partnership with Infosys Foundation USA, people learning to code in the US have an extra free resource to access. Discover more and how to get one of a thousand free kits here: **https://bit.ly/lxf282path**.



Everything you need to teach and learn coding.

CREDIT: Raspberry Pi Foundation

Home automation

Pi-powered, crowd-funded.

Home Assistant Amber is a home automation system powered by a Raspberry Pi Compute Module 4 that was fully funded after just a week on CrowdSupply. It's fitted with gigabit Ethernet (with PoE), a Zigbee 3.0/Matter module,an M.2 slot and custom enclosure with a heatsink, and is controlled via your smartphone. It's aiming for a May 2022 release. More at **https://bit.ly/lxf282home**.

The easy way to power your home... possibly.

CREDIT: Home Assistant Amber

Node-RED 2.0

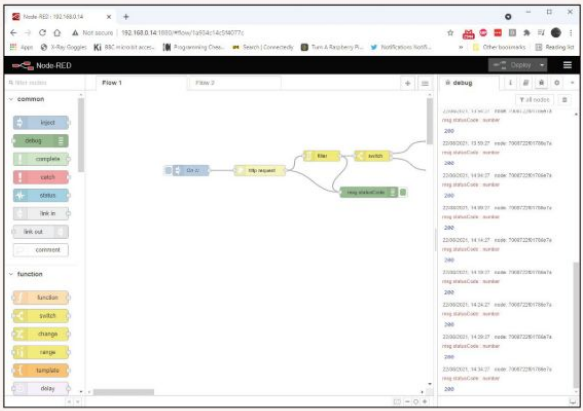
Les Pounder enjoys building things, and with Node-RED 2.0 he can construct the Internet of Things directly from his beloved Raspberry Pi.

IN BRIEF

Node-RED is an event-driven, node-based language designed for creating Internet of Things projects. The simple nature of the language belies the power just under the surface, and with a little time and patience we can soon make incredible projects. Node-RED 2.0 sees the language mature and improve, bringing new features for confident users, and improvements for new users.

Node-RED is a flow-based, event-driven programming language that uses nodes, similar in concept to Scratch blocks, but many times more powerful, which are joined to create a flow. Flows are triggered by an event – for example, checking the status of a website, then using the returned data to trigger a series of other events.

For Node-RED 2.0 we see support dropped for older Node.js releases, enabling the project to introduce a number of behind-the-scenes improvements. Front of house and we see an improved text editor, *Monaco*, which is the same as used in Visual Studio Code. For now this is an optional upgrade, and by default the traditional *Ace* editor is active, but all we need to change is one line in our config file and *Monaco* is ready to go.



The Node-RED user interface is simple and uncluttered. Underneath this simple UI there lies an extremely powerful IoT platform.

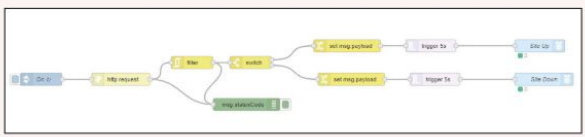
This addition alone is useful, with auto-complete, syntax highlighting and debugging all happening as we type. That said, there are some community created nodes that may not work as well, so make your choices wisely.

Triggering a flow can be done in many ways, but the most basic is an inject node, and for 2.0 we see the inject node has a new button in its dialog that will trigger the node without the need to come out of the edit dialog – useful for testing. Of course, the flow has to be deployed for the node to work, but it is a handy feature.

Site up or site down?

Another improvement is the RBE (Report By Exception) node, now renamed filter. The filter node can be used to block values from moving across the flow. For example, in our test we made a simple HTTP status code checked for **bigl.es**. If the site was up, we got a 200 status code, but if the site was down, or a bad URL, a 404 code was returned. In our flow we set it so that if the 200 status code was returned every five minutes, it only lit a green LED once. But if a 404 was returned, it constantly flashed the LED. The filter node is certainly a powerful and clever means to control a flow.

We installed Node-RED 2.0 using the official installer, a bash script from the Node-RED website, on a Raspberry Pi 4 1GB. The installer handles every facet of the base configuration. Once installed we run the *node-red-start* command and then open a browser, on the Pi or on another machine, to the IP address of the Pi, at port 1880.



Going with the flow, with the help of Node-RED.

The Node-RED user interface is clean. Palettes of nodes are on the left, in the centre is a space to build our flows, on the right is a combined information and debug window. The Raspberry Pi install also has extra nodes for controlling and reading the GPIO. There are installers for Debian and RPM-based distros, so you can deploy to your laptop and hack some Node-RED on the move.

Node-RED is a delight to use. We do need to wrap our heads around how it works – if we think of a traditional coding paradigm, then we'll soon hit a wall. Study the online resources to get a feel for how the nodes work. Once you get the hang of it, adding extra palettes of nodes is simple via the Manage Palette menu. Here you can add a plethora of nodes, including ready-made web UIs and industry standard IoT protocols. LXF

VERDICT

DEVELOPER: OpenJS Foundation
WEB: https://nodered.org
LICENCE: Apache Licence 2.0

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

Easy to install and use, Node-RED is the unsung hero of IoT and it works perfectly with Raspberry Pi.

» Rating 9/10

SCRATCH

Credit: https://scratch.mit.edu

Controlling the Pi's GPIO with Scratch

Les Pounder shows how we can use Scratch with simple electronics.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire.

YOU NEED

- **Any Pi model**
- **Latest Pi OS**
- **1x LED**
- **1x 220 Ohm resistor (RED RED BROWN GOLD)**
- **5x F2M jumpers**
- **Breadboard**
- **Push button**
- **1x PIR sensor**
- **Get the code: https://github.com/lesp/LXF282-Scratch-GPIO-Basics/archive/refs/heads/main.zip**

Scratch is meant for learning to code. It uses blocks to depict code and we can build sequences (algorithms) with relative ease. If something goes wrong we can take it apart and diagnose our code. As we've covered in previous issues, Scratch can also be used with the GPIO of a Raspberry Pi to learn electronics and in this tutorial we shall learn how to work with inputs such as buttons and a PIR sensor to control simple outputs such as LEDs.

Don't let the simplicity of this fool you. Adding the PIR sensor to our robot project (**LXF279**) and tweaking the code will enable our robot to detect an obstacle, and then move around it. But before we can do that we need to know the basics, so here we go!

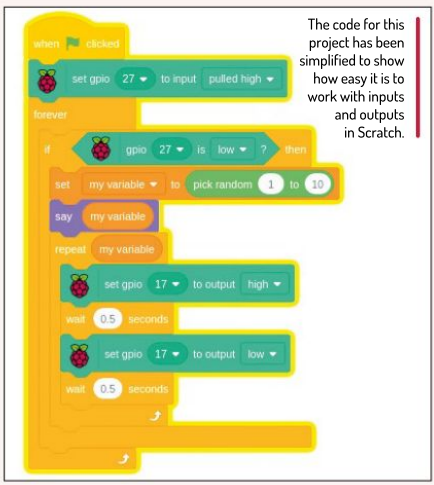
Scratch 3 should come pre-installed on your Raspberry Pi OS image, but just in case it's missing it can be installed from the main menu, under Preferences>Recommended Software. Scratch 3 is found in the Programming category; simply place a tick in the box and then click Apply to install. Once installed, Scratch 3 can be found in the main menu under Programming.

Open Scratch 3 – on first start Scratch may take a little while to open. We're going to assume that you have an understanding of how to code with Scratch. If not, we covered the basics in previous issues. Before we write any new code we need to click the blue folder icon in the bottom left of the screen to load the Extensions menu. From there select Raspberry Pi GPIO and a palette of new blocks is added to our code.

Reading and control

The Raspberry Pi's GPIO has at the most basic level two states. It can be on or off. We can replace on with True, High or 1, while off can be replaced with False, Low and 0. When a pin is high, it's active with voltage and current (power) and it can be used to power a device or as part of a circuit where we read the state of a pin and then react to changes.

In this project we'll use two GPIO pins: GPIO 17 for an LED and GPIO 27 for a button. The LED is connected using a breadboard, and has a 220 Ohm resistor inline between the short leg of the LED (cathode) and the GND connection. The long leg (anode) connects directly to GPIO17. The button connects to GPIO 27 and GND



The code for this project has been simplified to show how easy it is to work with inputs and outputs in Scratch.

directly. We use two pins, on the same side of the button as our connections.

In Scratch, go to Events and drag `When Green Flag clicked` and place it in the coding area. Scroll down to Raspberry Pi GPIO and connect the `set gpio 0 to input pulled high` block to the previous. Change the `0` value to `27`. Go to Control and drag `Forever` into the coding area and attach it to the previous blocks.

Also from Control, drag an `if..then` block and place it inside the forever loop. The if block is asking a question, and to do that we need to go to Raspberry Pi GPIO, drag `gpio 0 is high?` and place it inside the hexagon shaped slot on the block. Change the `0` to `27` and change high to low. When the button connected to GPIO 27 is pressed, its state changes from high to low as the button connects GPIO 27 to the GND pin (0V). When this happens it triggers the code inside the if test to run.

The code that shall be run starts by dragging `set my variable to 0` and placing it inside the if condition. We then go to Operators and drag `pick random 1 to 10` and place it on top of the 0 of `set my variable"`. This will now generate a random number between 1 and 10 each time

the button is pressed. We can have the cat sprite say the number (using a speech bubble) by dragging say hello from Looks and placing it under the previous block. Then drag another my variable from Variables and place it on top of hello . Give your code a test run, by clicking the Green Flag to start the code and then pressing the button on your breadboard. The cat should speak a number each time the button is pressed.

We have our input working and triggering the project – now let's add an output: the LED. Still inside our if condition we place a repeat 10 block from Control. Replace the 10 with another my variable block so that the loop will go round (iterate) for a random number of times, the same number as the cat speaks.

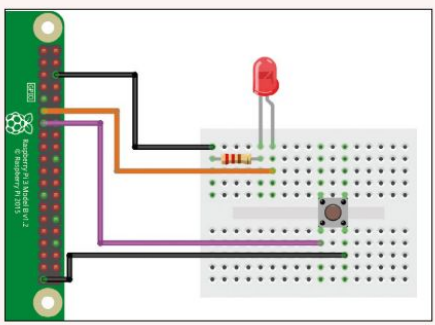
We want the LED to flash for the same number of times as the number stored in "my variable". To do this we'll turn the LED on (high) and off (low) with a short break between each change, giving us the appearance of a flashing LED. To control the LED we need to go to Raspberry Pi GPIO and drag two set gpio 0 to output high to the coding area. Place the first inside the repeat loop and change the values to read set gpio 17 to high .

Next, go to Control and drag wait 1 seconds and place it under the previous block. Change the one second delay to 0.5 seconds. Drag the second set gpio 0 to output high and place it under the previous block. Change the values to read set gpio 17 to low . The final block is another wait 1 seconds , placed under the previous block, and make sure to change the value to read 0.5 seconds. Click the Green Flag to start the code, then press the button to trigger the LEDs to blink for a random number of times.

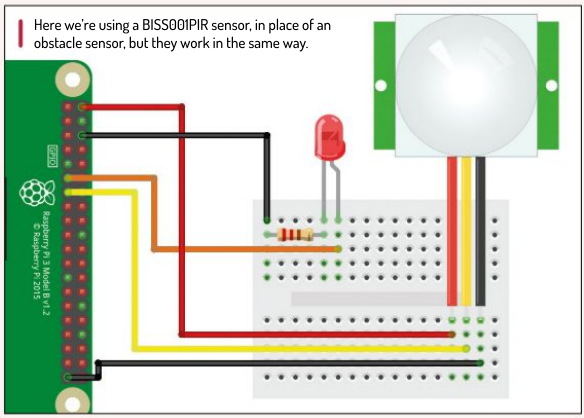
Using sensors

There's a limitation with Scratch for GPIO projects. We can only turn pins on and off and have no "fine" control such as Pulse Width Modulation (PWM), which can be used to control the brightness of LEDs and the speed of motors. This limitation is a result of the intended audience for Scratch rather than a technological one.

That doesn't mean that we can't use sensors with Scratch. In fact we're going to use a PIR (Passive Infrared) sensor with our code. Replacing the button with the sensor will give us a hands-free input. The PIR obstacle sensor has three pins. Black connects to any GND, Yellow connects to GPIO 27 and Red connects to the 5V pin of the Raspberry Pi. This sensor is powered by 5V, but the output pin is 3V meaning it's safe to use with the Pi. For other sensors always consult the datasheet before hooking it up because the Raspberry Pi GPIO could be damaged with 5V signals.



The button is an input, connected to GPIO 27. The LED is an output, connected to GPIO17 via a resistor.



Here we're using a BISS001PIR sensor, in place of an obstacle sensor, but they work in the same way.

Our code needs no changes for it to work. We just need to connect the pins of the sensor to the GPIO as mentioned above. Female to male jumper wires are exceptionally useful for this.

Once connected, click the Green Flag to start and then wave your hand about 15cm in front of the sensor. The back of the sensor should light up, triggering our LED to blink. If the sensor's LED doesn't light up, move your hand nearer until it does. We can adjust the trigger distance via a small screw on the rear of the sensor. Using a screwdriver, turn it clockwise/anti-clockwise until the correct distance is selected. LXF

QUICK TIP

PIR sensors come in many forms, the most common being a golf ball-shape. But they all work in the same manner. A good place to source these sensors is via the Cam Jam Electronics kit #1, which retails for around £5 and comes with multiple sensors and lessons.

» S3GPIO – ADVANCED SCRATCH

There's a version of Scratch which is still as easy to use, but it provides more options for electronics projects. S3GPIO was created and is maintained by veteran Raspberry Pi hacker Simon Walters. Installation is via a bash script, found at **http://simplesi.net/s3gpio** and once installed we run the application and find ourselves in the familiar Scratch interface. But from here we can control sensors, add-on boards and even NeoPixels (WS2812 RGB LEDs), which are notorious for their exact timing requirements.

S3GPIO runs via the web, so it's a little slow on older Raspberry Pis. A 3B+ and upwards should be fine. Using S3GPIO is a little trickier than Scratch. We first have to remember that the pin numbers refer to physical pin numbers, not the Broadcom GPIO reference that we typically use. So in our example GPIO 17 is pin 11, and GPIO 27 is pin 13. Yes this does get a little confusing, but if in doubt, remember that pin 1 is nearest the micro SD card, and pin 2 is just to the right of it. This means that we have two columns of pins, with 1 at the top left, and 2 at the top right.

Following Simon's interactive examples we can quickly recreate our project in S3GPIO and move beyond typical Scratch projects.

GPS RECEIVER

Setting atomic-clock precise time on your Pi

Sean Conway makes use of a Raspberry Pi Hat to pass some time, by setting up a GPS receiver to establish an accurate clock.

OUR EXPERT

Sean D. Conway After 40 years working with technology Sean uses Raspberry Pi projects to satiate his desire to explore electronics while having fun.

We all assume that our devices and computers all keep accurate date and time, but when was the last time you looked at the clock on your computer and asked, "I wonder if that time is correct?"

Often computers use the Network Time Protocol (NTP) from an internet time source to synchronise their time. In this tutorial, we're going to examine a system for establishing accurate time without an internet connection. We will assemble a global positioning system (GPS) receiver using a Raspberry Pi and take advantage of the accurate time signals that are broadcast by GPS satellites.

Establishing a GPS reference time source is a complex task so we'll break the task down into some manageable parts. First, we'll install an Adafruit GPS Receiver HAT and configure the software. Second, we'll examine the output signals provided by the receiver and take advantage of the GPS supplied tools to tweak the receiver for obtaining accurate time. Finally, we'll configure a daemon service that uses the GPS signals to ensure the computer's system time is accurate.

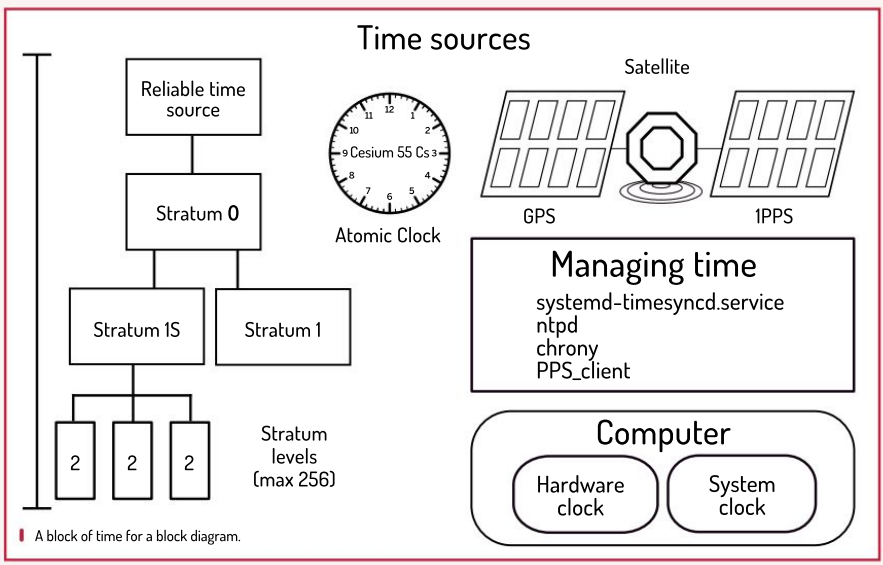
A reliable time source used as a reference could be an atomic clock or the master clock on a GPS receiver. NTP uses a stratum levels to organise the hierarchy in a time servers network. A Stratum 0 device is directly connected to a reference (reliable) time source. Stratum 0 may either be an atomic clock or a GPS receiver using the clock signals broadcast by GPS satellites.

Stratum 0 devices can't distribute time over a network directly. They're linked to a Stratum 1 time server that will distribute time to Stratum 2 servers or clients connected. The lower the Stratum number, the less timing accuracy and stability of time signal.

Computers worldwide use NTP to synchronise their times with internet-standard reference clocks via a hierarchy of NTP servers. *systemd-timesyncd* is the Systemd service that can synchronise the local system clock with a remote NTP server. Applications like *ntpd*

YOU NEED

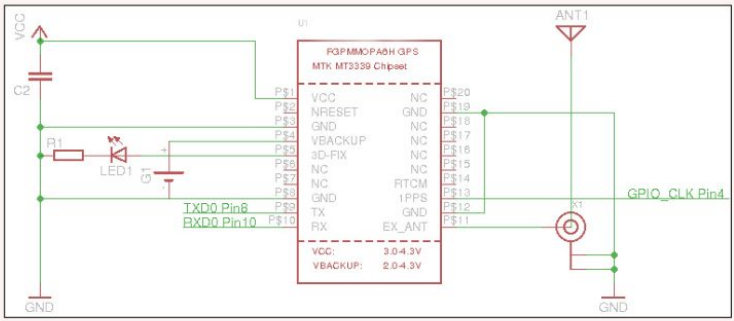
- Raspberry Pi 3B+
- 8+GB SD card
- Adafruit Ultimate GPS HAT Mini Kit
- Raspberry Pi OS Lite (4/3/2021)
- MT3339/ PA6H GPS software tools
- PPS-Client v2.0.3



A block of time for a block diagram.

and *chrony* can also obtain time from stratum time servers to manage time. GPS reception doesn't require such a internet access network. GPS signal paired with a daemon like *PPS-Client* can be used for managing time.

For this project the Adafruit Ultimate GPS HAT adds a GPS receiver to our Raspberry Pi. The receiver and Pi hardware combination can determine location and precision time. The HAT provides a real-time clock (i.e. hardware clock) with a back-up battery for timekeeping when the Pi is turned off. The Pi had no hardware clock.



The beating component at the heart of the GPS receiver HAT.

Locate your HAT

The HAT GPS Module comes with a built-in antenna. To improve satellite reception for this project, an external active antenna was installed. The GPS detects this antenna and switches over to it once it's installed.

The GPS HAT EEDATA/EECLK communication channel is connected to the Raspberry Pi serial port (TXD0, UART-pin 08, RXD0 UART-pin 10). Data output from the receiver is 9600 baud by default. The GPS module also outputs a pulse per second (PPS) signal for improving time synchronisation. The PPS output connects to pin 07 (GPIO4) of the Raspberry Pi.

Now that we have the GPS HAT module installed, let's tailor the Pi to support the module. Before making operating system (OS) changes, ensure all the software in the OS is up to date. To do this, issue this command from a command-line interface (CLI) on the Pi:

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

When the OS update is complete, open the Raspberry operating system configuration tool *raspi-config*, to complete the configuration changes to establish communication between the GPS HAT and the Pi. Using the *raspi-config* menu, select Interface Option>P6 Serial Port, disable the login shell and enable the serial interface. Exit the tool when complete.

The OS configuration tool removes the heavy lifting of modifying the **/boot/config.txt** and **/boot/cmdline.txt** configuration files. Letting a software tool make changes is efficient for an administrator. Here is the insight into what the tool changed in the files.

```
/boot/config.txt
enable_uart=0
```

after

```
enable_uart=1
/boot/cmdline.txt
```

before

```
console=serial0,115200 console=tty1
root=PARTUUID=be89cf0a-02 rootfstype=ext4
elevator=deadline fsck.repair=yes rootwait
```

after

```
console=tty1 PARTUUID=be89cf0a-02 rootfstype=ext4
elevator=deadline fsck.repair=yes rootwait
```

Unfortunately, *raspi-config* doesn't take care of all the configuration changes required to support the GPS HAT module. Using your favourite text editor, add the following lines to the bottom of the **/boot/config.txt** file.

```
#PPS from HAT pin
dtoverlay=pps-gpio,gpiopin=4
```

The option enables the OS kernel module to provide **/dev/pps#** device support through a GPIO pin connected to the PPS source. A restart of the Pi will be needed for the changes to take effect.

The *systemctl* command-line tool can list the services that are currently running. *systemd-timesyncd* is a service that's used on the Pi to synchronise the local system clock with an internet NTP server. It steps the system clock with small offsets to synchronise time.

Before adding the PPS software tools that will take advantage of the GPS signals the receiver is providing, let's remove the default services that are managing time on the Pi. Stop the service and disable the service when the system boots using the command line.

```
$ systemctl list-units --type=service –all
$ systemctl stop systemd-timesyncd.service
$ systemctl disable systemd-timesyncd.service
```

From the CLI on the Raspberry Pi, issue the following commands to install software packages and configure the serial port communication protocol characteristics.

```
$ sudo apt install pps-tools python-serial libncurses5-
dev libncursesw5-dev
$ sudo stty -F /dev/serial0 raw 9600 cs8 clocal -cstopb
```

QUICK TIP

Add precision time and location to your Pi at https://learn.adafruit.com/adafruit-ultimate-gps-hat-for-raspberry-pi.

» GPS SENTENCE DATATYPES

The National Marine Electronics Association (NMEA) has a standard for the formatting of GPS information. ASCII text, comma delimited lines of data transmitted from the GPS satellites are referred to as sentences. The first word of each sentence is prefixed with **$GP** (i.e. global positioning) and defines the data type. For a more detail description about these or other NEMA data types see **http://aprs.gids.nl/nmea**. Some of the standard NMEA-0183 datatypes include:

Datatype	NEMA description
GPGGA:	Time, position and fix
GPGSA:	GPS receiver mode, satellites used in the position solution
GPASV:	Number of satellites in view, ID numbers, elevation, azimuth
GPRMC:	Time, date, position, course and speed
GPGZDA	Pulse per second (PPS) time message

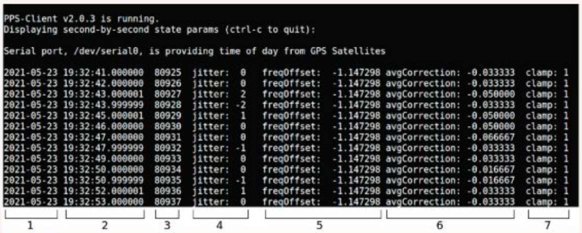
The *pps-tools* software package will enable us to examine the PPS data provided by the GPS receiver. Serial interface support for Python is provided through the *python-serial* install as well as some curses support for test software we'll use later. *stty* is a Linux command-line tool that's used to change terminal device characteristics. The command is setting the port **/dev/serial0** to a baud rate of 9,600, eight-bit data, modem control off and one-stop bit. Now that the serial port has some data formatting, let's look at the **serial0** output.

```
$ cat /dev/serial0
```

The GPS receiver provides output on the serial port depending on if a "fix" is established.

Examining the outputs, the sentences with multiple commas are missing data because a satellite fix isn't established at the receiver.

GPS receivers provide data if there's a sufficient signal level (>25dBHz) [a fix] from a minimum of four satellites for about a minute. If the signal level degrades because the view is obstructed or a satellite drops out of view at that minute, the GPS receiver has to start over again.


PPS-Client v2.0.3 is running.
Displaying second-by-second state params (ctrl-c to quit):
Serial port, /dev/serial0, is providing time of day from GPS Satellites
2021-05-23 19:32:41.000000 80925 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:42.000000 80926 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:43.000001 80927 jitter: 2 freqOffset: -1.147298 avgCorrection: -0.050000 clamp: 1
2021-05-23 19:32:43.999999 80928 jitter: -2 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:45.000001 80929 jitter: 1 freqOffset: -1.147298 avgCorrection: -0.050000 clamp: 1
2021-05-23 19:32:46.000000 80930 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.050000 clamp: 1
2021-05-23 19:32:47.000000 80931 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.066667 clamp: 1
2021-05-23 19:32:47.999999 80932 jitter: -1 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:49.000000 80933 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:50.000000 80934 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.016667 clamp: 1
2021-05-23 19:32:50.999999 80935 jitter: -1 freqOffset: -1.147298 avgCorrection: -0.016667 clamp: 1
2021-05-23 19:32:52.000001 80936 jitter: 1 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
2021-05-23 19:32:53.000000 80937 jitter: 0 freqOffset: -1.147298 avgCorrection: -0.033333 clamp: 1
1 2 3 4 5 6 7

Columns of significance as PPS client establishes a hold over time.

Col	Description
1	Date of the rising edge of the PPS signal.
2	Time of the rising edge of the PPS signal.
3	Sequence count: total number of PPS interrupts received since starting.
4	Jitter: time deviation in microseconds since the reception of the PPS interrupt.
5	freqOffset: correction offset in parts per million applied to the system clock.
6	avgCorrection: parts per million that's applied once per minute to the clock.
7	Clamp: hard limit applied to the time error to create the time correction.

» GPS RECEIVING DETAILS

Most people are concerned with time. Some of us are even obsessed with trying to create time. If you master your time, you can get more done and do more things you enjoy. As Theophrastus said, time is the most valuable thing a man can spend.

GPS receivers depend on a constellation of satellites and ground stations to compute their position and time. The satellites constantly transmit position and time data over radio frequency ranging from 1.1 to 1.5GHz. Using the data provided from a minimum of four satellites and geometry process called trilateration, the GPS receiver module can accurately calculate its position and time.

The data sent down to earth from satellites contains information that GPS receivers use to determine their time and position. If you want more information on how GPS receivers do the math, check out volume 1 of *GPS Fundamentals* by Dan Doberstein (**https://cdn.sparkfun.com/datasheets/Sensors/GPS/fundamentals_of_gps_receivers-v1.pdf**).

It can prove difficult indoors for the GPS receiver to gain a fix if four satellites don't deliver a strong-enough signal in the required time window. Outdoors, and the problems that degrade received signals are reduced.

Another signal sent from the GPS satellites is pulse-per-second (PPS). The PPS signal is emitted within 10ns of the beginning of each GPS second. This signal along with the GPS time is used to adjust the system clock to greater accuracy.

A PPS source can be connected to a serial port (the Data Carrier Detect pin) or to a parallel port (ACK-pin) or to a special CPU's GPIOs (this is the common case in embedded systems), but in each case when a new pulse arrives the system must apply a timestamp and record it. Execute the following command from the CLI to confirm a PPS signal is being received.

```
$ sudo ppstest /dev/pps0
```

The GPS Receiver's sync to the GPS clock system depends on the quality of the fix, how long the receiver has had a fix, and the group of satellites used by the satellite. To assist the GPS receiver in getting a fix on satellites, the vendor provides ephemeris data (location of the satellites) as Extended Predicted Orbit (EPO) data that can be uploaded to the receiver. Using EPO data, the requirement for a stronger signal is reduced.

From the CLI issue, the following commands:

```
$ sudo wget https://github.com/f5eng/mt3339-utils/archive/gps.zip
$ unzip gps.zip
$ cd mt3339-utils-gps/
$ sudo cp gpsinit_* /etc/
```

The command series downloads and installs a software toolset for the GPS receiver. After changing directories, we copy the configuration files to the **/etc** directory. Now execute the following command. (The command will need to be executed twice.)

```
$ sudo ./gpsinit -s 115200 -f gpsinit_reset.conf /dev/serial0
```

The first time the command is executed, the output includes error messages. The command output ends with a `Done`. Run the same command a second time. The second time the command is executed, the output doesn't contain error messages. In addition, the software establishes an FTP connection and downloads the EPO data files that are uploaded to the GPS Receiver module. The command output finishes with a `Done`. Looking at the output from the command, it appears another Done is provided on completing the command the second time.

From the CLI check the output of the serial interface:

```
$ cat /dev/serial0
```

The GPZDA:PPS timing message (synchronised to PPS) data type sentences weren't available at the serial output until the EPO data was provided to the GPS receiver. From the vendor receiver tools directory, execute the following command:

```
./gpsstatus /dev/serial0
```

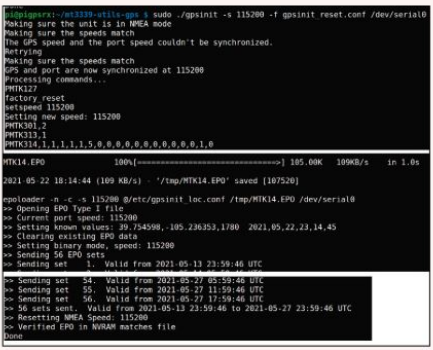
The vendor command-line tool uses the GPS data types to produce the formatted output (*see opposite*).

A The GPS receiver determining the lat. and long. coordinates. Using the coordinates in Google Maps the user could confirmed the location of the author's house.

B The list of the satellites tracked by the receiver.

C A confirmation of the EPO data that was loaded.

D The data types provided through the serial port.



The second output iteration of the command loads the data.

Menu items for the tool appear across the top of the display. Q quits the display.

Let's install the *PPS-Client* daemon that synchronises the system time to the GPS time using the PPS signal. The daemon uses the *pps-gpio* driver provided in the Linux kernel that was installed early in the **/boot/config.txt** file.

```
$ sudo apt install git
$ git clone --depth=1 https://github.com/rascol/PPS-Client;
$ cd PPS-Client/
$ ./create-backups
$ make
$ sudo make install
```

After you install git support, the *PPS-Client* repository is installed. Before compiling the daemon, the software requires a backup. To establish the configuration file use a text editor and ensure the following changes are applied from **/etc/pps-client.conf**. A description of each parameter is documented in the configuration file.

```
error-distrib=enable
exit-lost-pps=enable
serial=enable
serialPort=/dev/serial0<--change the default reference
ppsdevice=/dev/pps0
```

The *PPS-Client* daemon provides the time keeping precision accuracy of two microseconds on Linux processors with operating systems that provide the Linux *pps-gpio*. *PPS-Client* uses the PPS hardware signal from a GPS HAT present on GPIO 4.

To install the *PPS-Client* as a daemon, type:

```
$ sudo pps-client
```

The daemon can be stopped with this command:

```
$ sudo pps-client-stop
```

To run a continuous status display of the *PPS-Client* daemon, issue the following command:

```
$ pps-client -v
```

The daemon continuously generates second by second timekeeping parameters and saves the data to memory. The client tools display the data that claims to discipline the system clock using the GPS-supplied PPS signal. The *PPS-Client* daemon output data stream contains 10 columns.

The jitter column is the one to watch. The value is the fractional offset of the PPS signal according to the system clock. It can take over 20 minutes for the *PPS-Client* daemon to reduce the jitter value to around 1ms.

The jitter value at startup is in the order of 100K microseconds. The jitter value decreases after each sequence count as the controller locks to the PPS signal.

When the jitter column has reached signal digit values, watch the rising edge of the PPS signal shown in Column 2. The clamp value in Column 7 shows the maximum +/- microsecond time correct in applying to the system clock. The system clock is being synchronised to the GPS PPS signal with a precision of one microsecond.

The daemon may have to restart several times (i.e. sequence count will reset) to reduce the jitter offset after a cold start. The *PPS-Client* is limited to time changes it can make using a system function that prevents the daemon to less than 500 microseconds of change in each second. A *PPS-Client* daemon will continue to run until the system is rebooted or until stopped. The *PPS-Client* can be optioned to run as a Systemd service. From the command line, issue the following commands:

```
$ sudo systemctl enable pps-client
$ sudo systemctl start pps-client
```

The *PPS-Client* documentation suggests the daemon can be thought of as two conceptually different components: a feedback controller and a feedforward compensator. The purpose of the feedback controller is to adjust the system time second by second. A feedforward compensator provides the compensation delay values.

Since the daemon writes data values to memory, this data can verify that system time is being controlled and that the client is active. Execute the following command twice in succession to generate an output:

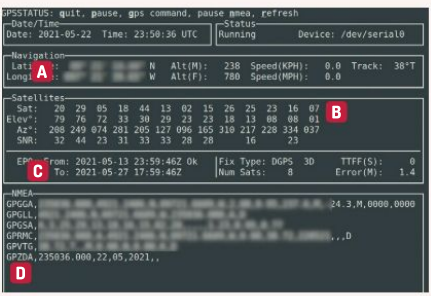
```
cat /run/shm/pps-assert
```

The timestamp data is displayed in seconds to the nearest microsecond. If the two values were equal, it would be an indication the daemon was not running:

```
1621822332.000001#86896
1621822333.999999#86898
```

Well, we're bumping up against the end of time for setting up a GPS receiver to use a PPS client to establish accurate time. The tutorial explored some of the GPS signals transmitted by satellites. The knowledge enables us to set up our own GPS receiving station and examine the messages sent to Earth.

To obtain those messages we needed to enable the receiver to acquire a fix by providing it some ephemeral data. Using those messages our receiver was able to generate a signal, which was then used to establish PPS client to provide accurate time on the Raspberry Pi. And... our time is up! LXF



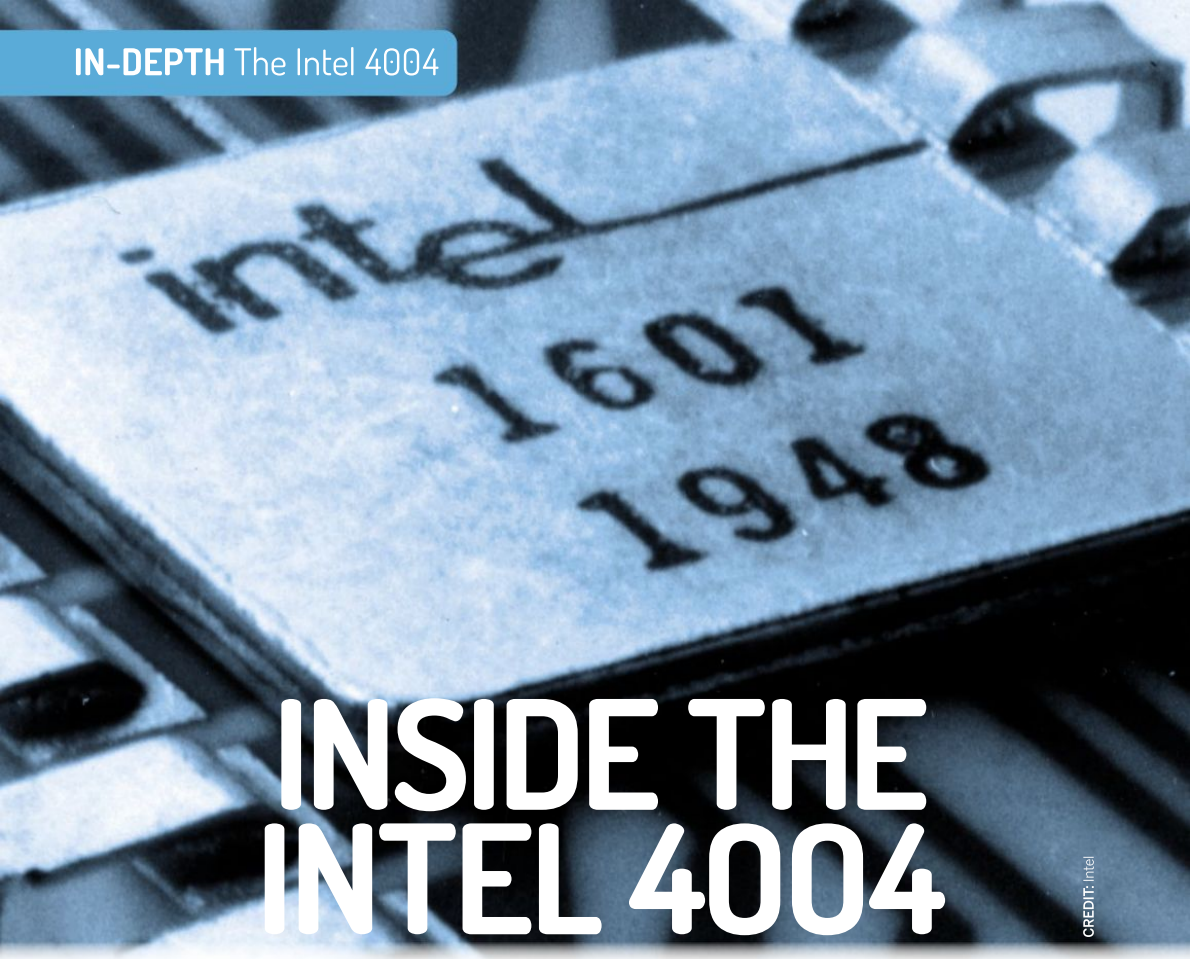
Command line output of gpsstatus command when the fix is in place.

QUICK TIP

In order for GPS to get an accurate fix it needs to know time, location of itself and other satellites. Hey that's strange... it's suppose to tell us those things https://github.com/gtjoseph/mt3339-utils.

QUICK TIP

The PPS-Client daemon is a fast, microsecond accuracy Pulse-Per-Second system clock synchroniser for application processors that synchronises the system time to GPS time from the PPS signal. See https://rascol.github.io/html.



INSIDE THE INTEL 4004

CREDIT: Intel

Mike Bedford shows you how to use an emulator to get hands-on experience of the microprocessor that started a revolution.

This issue we're celebrating a significant 50th anniversary that's been overlooked by many. That milestone was the announcement by Intel of "a new era of integrated electronics" as it was heralded in the November 1971 issue of *Electronics News*. The advertisement was for the Intel 4004, the world's first microprocessor.

It's no surprise that this breakthrough didn't make it into the newspapers. Somewhat more surprising – given that computing and microprocessors are now synonymous – is that it seems much of the computing industry barely gave it a second glance, either. After all, compared even to the primitive computers of the early '70s, it would have looked like a toy – and we'll see something of its basic nature later on.

Indeed, few would have envisaged that a microprocessor would ever become the basis of general-purpose computers. It was designed to do a particular job, namely to power an electronic calculator, and even when Intel decided to promote it beyond these humble beginnings, most engineers considered it as just a replacement for discrete digital electronics chips. But no matter how basic its impact at the time, it started an industrial trend that continues to drive innovation today.

To explore its primordial design we're taking a practical look at its architecture through programming it in assembly. As our first program we'll investigate a simple example of adding numbers in two memory locations and storing the result in a third. So we're talking of what could be done in Python with a single statement such as `sum = value1 + value2`. Turning to early microprocessors – which were more often programmed in assembly than today's chips – we inevitably needed more instructions.

For example, we saw the following 8080 code needed to carry out the operation when we looked at the Altair 8800 back in **LXF278**.

`VAL1` , `VAL2` and `SUM` having been defined as memory addresses via assembler directives:

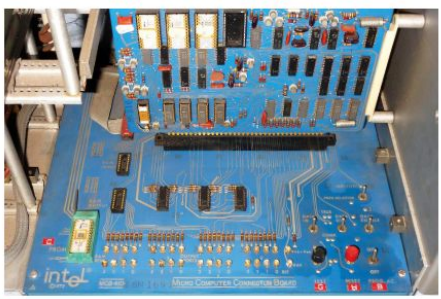
```
LDA   VAL1
MOV   B,A
LDA   VAL2
ADD   B
STA   SUM
```

This might be a significant increase on Python's single instruction, but that simple addition took yet more on the 4004, as the following code shows:

```
LDM   $00
DCL
FIM   0P %00000000
SRC   0P
RDM
FIM   0P %00000010
SRC   0P
CLC
ADM
FIM   0P %00000100
SRC   0P
WRM
```

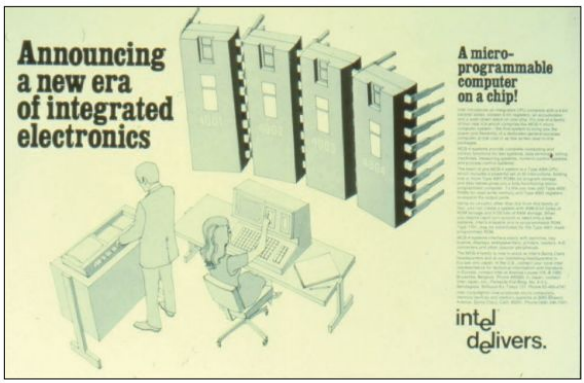
We'll describe what each of the instructions does shortly, but first a few words on the 4004's memory architecture, which is one of the main reasons for 4004 programs being so verbose.

Back in 1971, most integrated circuits were housed in small dual-inline packages with typically 14 or 16 pins. In time, it would become clear that microprocessors needed more pins. Indeed, in just a few years they'd be housed in 40-pin packages and today's chips might have a few thousand connections. The 4004 was housed in a 16-pin package and the upshot of its shortage of pins was that memory addressing was a complicated process. First, a memory bank had to be selected, and then it was necessary to select one of four RAM chips within the bank, one of four registers within that chip, and one of 16 four-bit characters within the register. This requires several instructions to be executed to set all this up, before issuing an instruction to access memory.



CREDIT: www.intel-vintage.info

This SIM4-01 was probably the closest thing to a 4004-based PC that ever existed, but it was really just a development aid for engineers.



CREDIT: Intel

This ad for the 4004 also showed the rest of the chipset, the 4001 ROM, the 4002 RAM, and the 4003 Shift Register.

So now to the instructions in our first program. The `LDM` instruction loads the value 0 into the accumulator and the `DCL` then uses the least significant three bits of the value in the accumulator to select the memory bank for future accesses (in this case, bank 0 of 8). The `FIM` instruction writes the value 0 to the eight-bit register pair 0P, which comprises the two four-bit registers R0 and R1. This is then used in the `SCR` instruction to define the memory chip, bank and character address for subsequent accesses.

We have shown the value in `FIM` instructions in binary to clarify that the value represents the chip number (most-significant two bits), the register within that chip (next most-significant two bits), and the four-bit character within that register (least-significant four bits). Having set up the bank, chip, register and four-bit character, the `RDM` reads the value from the defined location into the accumulator. The next two instructions sets up a different memory location for access.

INSIGHT

It has been reported that the Intel 4004 was used in the Pioneer 10 space probe launched in 1972 to explore Jupiter. This is confused with that the spacecraft carried a 4004 as an artefact, to demonstrate levels of human technology.

» INVESTIGATING BIT-NESS

The 4004 is considered a four-bit processor, but what do we mean when we talk about a processor's "bit-ness"? It has been described as the width of all the processor's registers and busses, but it's not at all that simple. For example, we've seen that the 4004 only had four-bit registers so it could only do four-bit arithmetic, but if that extended to the address bus, it would only have been able to access 16 memory locations.

The address bus continued to be wider than the data bus in most processors up to 16 bits, and we were told that the main reason for migrating to 64 bits was to access more than the 4GB limit imposed by 32-bit processors. It didn't increase to 64 bits, though – after all, few people would need 18.4 exabytes of memory, which is several thousands time the memory capacity of the world's fastest supercomputer.

The width of the data bus, internal registers and ALU have also increased and is normally the same as the quoted "bit-ness", but not always. For example the 8088, as used in the original IBM PC, was a variant of the 16-bit 8086. Yet while all the internals were 16 bits wide, to permit it to be used with eight-bit memory, it had an eight-bit external data bus. This meant that two memory access were needed to read a 16-bit value into a 16-bit register.

It can go the other way, though, with many of today's processors having 128- and 256-bit instructions, enabling operations on multiple shorter values to be carried out in parallel.

INSIGHT

The word byte is commonly used refer to eight-bit values, even in these days of 64-bit processors. However, you might not be familiar with the word nibble, which is a four-bit value. If there had ever been two-bit processors, we wonder whether the term lick would have been coined?

We're about the use the `ADM` instruction, which adds the value in the memory location to the value in the accumulator. Yet because that instruction also adds in the carry bit (most processors have two different `ADD` instructions – one with carry and one without) it's necessary to clear the carry bit, which is what `CLC` does. The two instructions following the `ADD` select a third memory location for access, and the final instruction, the `WRM`, writes the result of the addition in the accumulator to the selected memory location.

Emulating 1971...

We chose to feature a well-respected online emulator of the 4004 because the state of the chip can be seen at a glance. The emulator is at **http://e4004.szyc.org**. In addition to the emulator, there's also a disassembler and an assembler hosted here, the latter of which will be particularly useful in your experiments with the 4004.

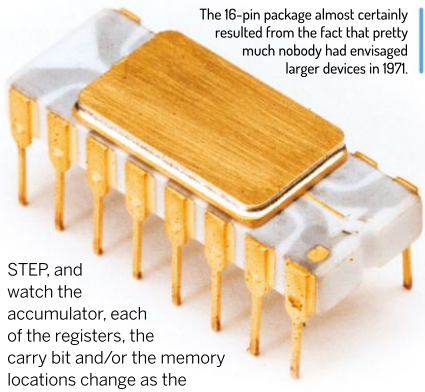
We suggest trying out our first 4004 program in the emulator. Because we've presented the code as assembler instructions, you'll first need to assemble it to generate the opcodes that you'll enter into the emulator. So, head over to **http://e4004.szyc.org** and click Assembler. Enter the code into the 'source code:' window and click 'generate code'. If it reports an error then correct the source code; otherwise select the code in the 'object code:' window and copy it. Now return to the home page and select Emulator, paste the object code in the 'ROM block:' window and click LOAD. Immediately you'll see the first instruction, waiting for you to execute it, displayed at the bottom-left of the '4004 CPU:' window.

From small acorns mighty oaks grow: the 4004 was originally designed to power this desktop calculator.

CREDIT: Christian Bassow

Before executing the code, though, you need to write the two values to be added into bank 0, chip 0, register 0, characters 0 and 2. You can do this in the '4002 RAM:' window, but note that the characters 0 to F are arranged left to right, which might be the opposite to what you'd expect. To change a value, click it and you'll be prompted for a new value – enter this as a single hexadecimal digit.

With everything set up, it's time to execute the code and we recommend that you single-step through it by repeatedly clicking STEP, and watch the accumulator, each of the registers, the carry bit and/or the memory locations change as the instructions are executed.



The 16-pin package almost certainly resulted from the fact that pretty much nobody had envisaged larger devices in 1971.

CREDIT: Intel

To high-level language programmers, needing 12 instructions to add together two numbers will seem incredible, but there's more. As a four-bit processor, the 4004 could only work on four-bit numbers, which means integers in the range 0 to 15. As a universal processor, it could add together longer integers, but at the cost of yet more instructions. We'll leave you to figure it out yourself, but we suggest you enhance the previous code so it works on eight-bit values.

The two numbers we added were at memory locations 0 and 2, and the result was written to address 4. We used alternate addresses so pairs of adjacent locations could hold eight-bit numbers as two four-bit values. Essentially, you'll duplicate your existing code, but make a few changes to the duplicate. You can omit the `LDM` and `DCL` instructions because we'll be using the same memory bank, and you'll need to change the `FIM` instructions to address characters 1, 3 and 5, instead of 0, 2 and 4. You'll also need to delete the `CLC` instruction because this time you'll want the carry bit to be added into the second `ADD` instruction, because it'll have been set by the first `ADD` if the result exceeded 15.

4004 instructions

The `ADD` instruction is one of the 4004's group of arithmetic instructions, although the list isn't too extensive. To that we can add the subtract, increment and decrement instructions, but that's about all. Most

» THE NEED FOR SPEED

The 4004 might have required lots of instructions to carry out the simplest of jobs, but the speed also depends on the clock speed and architecture. The clock speed was 740kHz, so just a bit below 1MHz and several thousand times slower than today's latest and greatest.

Clock speed alone doesn't tell the full story. With initiatives such as multiple cores, and superscalar architectures in which each core contains several execution units, today's processors can execute many instructions in parallel. Prior to these innovations, there had been moves to execute a single instruction per clock cycle, but this came well after the 4004. To quantify this, the 4004's single-byte instructions required eight clock cycles per instruction, and this doubled with two-byte instructions. The processing speed was less than 92 KIPS – that's thousandths of a MIPS.

Benchmarking isn't an exact science, because it'll never be representative of all real-world applications. In theory, though, it measures performance so it takes into account the sophistication of a processor's instruction set, its clock speed and its internal architecture.

The figures make interesting reading. It transpires that the 4004 is over eight million times slower than a top-end processor intended for desktop applications. That's some rate of progress in 50 years, and we've surely not finished yet.

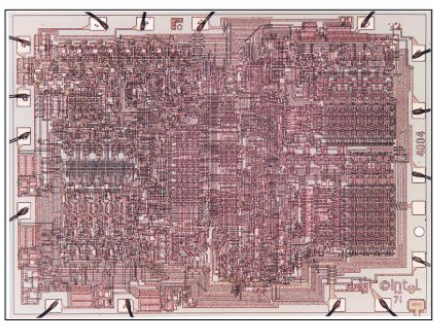
notably, there's no multiply instruction, although the "rotate accumulator left through carry" instruction multiples by two. This deficit isn't at all surprising since many eight-bit processors, including some that were in widespread use a decade after the 4004, were also devoid of multiply. This meant that multiplication had to be carried out using a program which, in the general case, employed multiple additions and shifts.

We're going to implement multiplication as our second example, but to keep things simple we'll only use additions, which would have been much slower. We're going to further simplify it by only multiplying four-bit positive numbers, which limits us to computing results that don't exceed 15. And as our final simplification, since we've already seen how to access memory, we're going to multiply two numbers that actually appear in the code, although you can change those numbers before assembling the code. However, it does enable us to see a couple of flow control instructions – the code appears below:

```
        LDM     4
        XCH     R0
        LDM     3
        CMA
        XCH     R1
        CLB
        JUN     TEST
ADLOOP          ADD     R0
TEST    ISZ     R1 ADLOOP
LOOP    JUN     LOOP
```

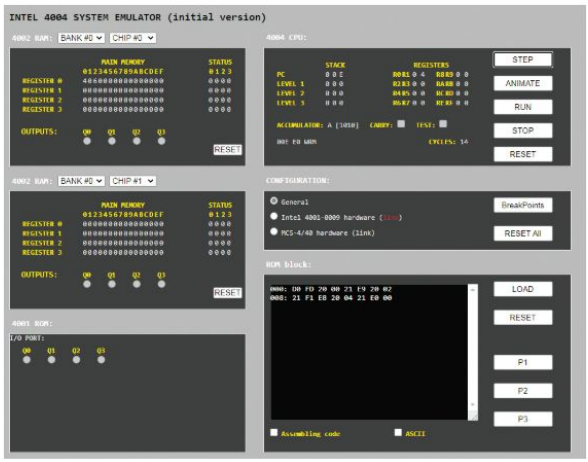
The `LDM` instruction loads the value 4 into the accumulator and the `XCH` exchanges the values in the accumulator and the specified register, R0. Now we load the value 3 – the other number we're going to multiply – into the accumulator and the `CMA` instruction complements it. Specifically, it generates the 1s complement, which involves flipping each of the bits between 0s and 1s.

We execute another `XCH` instruction, this time with R1, so after this instruction we have one of the numbers to be multiplied in R0, and the complement of the other in R1. Next is the `CLB` instruction that clears both the accumulator, which we'll be using to store the running total, and the carry bit, so that the subsequent `ADD` instruction won't inadvertently add in the carry bit. The `JUN` instruction is an unconditional jump and, specifically, it jumps into the loop after the `ADD` instruction. This is because the value in R1 (the complemented value), which we're using as the loop counter, will cause the loop to be executed one too many times. Then the `ADD` instruction – which isn't executed the first time round the loop – adds the value of R0 to the running total which is in the accumulator.



CREDIT: Intel

A total of 2,300 transistors might not be a lot, but it's surely a testimony to 4004 creator Federico Faggin that so much could be done with that modest silicon real estate.



This emulator, at http://e4004.szyc.org enables you to assemble and execute 4004 code while watching the contents of memory and the chip's internal registers.

The final instruction in the loop, the `ISZ`, increments the value in R1 and loops back to the start of the loop unless the result of the incrementing is zero, in which case the multiplication is complete and execution continues with the next instruction. That final instruction, which just loops back to itself, has been added so it'll be obvious when the multiplication is complete, because this instruction will repeatedly show in the emulator as the next one to be executed.

Beyond the instructions

Our main emphasis here has been the instruction set, but the ways in which the 4004 differed from its modern-day successors goes far beyond that. While it might have sounded a lot in 1971, the 4004's 2,300 transistors pales into insignificance compared to the estimated seven billion of a top-end Core i9, and the tens of billions boasted by some GPUs. This increase has been fuelled to no small extent by decreasing feature sizes, and here we'd have to point out that the 4004 was manufactured using a 10µm process, which is up to a couple of thousand times larger than today's leading-edge 7nm or even 5nm processes. But the transistor packing density is roughly proportional to the square of the process size, so manufacturers can now pack a million times more transistors onto square millimetres of silicon.

Finally we come to power consumption, which has become a much greater concern over the 50 years of the microprocessor age. The 4004 consumed 1W, but it was over eight million times slower than today's top-end chips. Had electrical efficiency remained the same, today's latest and greatest would consume 8MW. That's about as much as 4,000 domestic electrical ovens. So, if you still need convincing of the benefits of 50 years of development, bear in mind that you could have been paying £9,000 per day to use your PC. LXF

INSIGHT

Today's mainstream processors employ the Van Neumann architecture, so the program and data reside in the same memory space. The 4004 had a Harvard architecture – it had separate address spaces for both code and data.

BORG

Create efficient backups

Backups aren't just a safety net for Trekkies such as **Shashank Sharma**, but a way of life. But they can take up quite a lot of space if not done right...

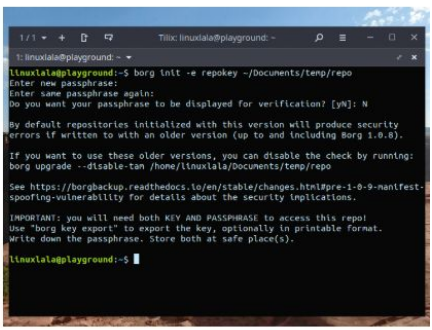
OUR EXPERT

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

Most backup strategies lead to a lot of wasted space because you invariably end up making copies of the same data over and over again. In essence, data deduplication is the process of comparing chunks of data, and retaining only a single unique copy, thereby saving disk space. When deployed as a data backup strategy, this means that only changes since the last backup are stored.

Backup tools for Linux are as prolific as browsers, video players or even text editors, *BorgBackup*, or just *Borg*, is a deduplicating backup utility. Released under the BSD license, the project was forked from Attic, a popular backup utility, in 2015. Thanks to the data deduplication technique, *Borg* is suitable as an everyday home or even enterprise backup solution, with a host of features that make it a breeze to use.

Since *Borg* is only interested in changes since your last backup, the entire process is fast and efficient. Even better, *Borg* also supports compression, which further limits the amount of disk space eaten by backups. Finally, *Borg* also supports client-side encryption, so you can safely encrypt backups before they are transmitted to remote storage locations. Apart from all that, *Borg* backups can be mounted as a filesystem. This means that you can use a file manager to examine the contents of the backup, and restore only specific files.



The borg init command is used to initialize an empty repository. In this example, we've created a repository called repo in the ~/Documents/temp directory.

The Borg are good!

No self-respecting Trekkie ever thought of Borg as good, but here at **LXF** towers, we are pleased to break the mould. You can gauge the popularity of *Borg* from the fact that it's available in the software repositories of most popular desktop distributions. The project's website, however, cautions that the repositories might not always carry the latest version. Thankfully, the current release, version 1.1.17, is available in most popular distributions such as Fedora, Ubuntu, Debian, Mageia, Arch, openSUSE Tumbleweed, and others.

The project's website has a list of distributions, and the version in their repositories. Run the `sudo pacman -S borg` command to install *BorgBackup* on Arch/Mageia. You can similarly run the `dnf install borgbackup` command to install it on Fedora.

As can be expected, the project has a long list of dependencies, so if you decide to compile it from source, please carefully follow the installation instructions on the website. Just the thorough focus on installation make *Borg* one of the best documented command-line tool we've encountered.

The first step is to create a backup repository. You can think of a repository as a filesystem that's home to

» COMPRESSED ARCHIVES

When creating a new archive, you can also use the `--compression` or the `-C` command option to create a compressed backup archive. *Borg* defaults to using LZ4, which is fast but offers little compression.

Borg also supports other compression algorithms such as zstandard, zlib and lzma. You can use a mix of compression algorithms for different archives within the same repository. This is because deduplication is done on the source data chunks.

With the exception of the default LZ4, all other algorithms expect a compression level. For zstd, the compression levels range from 1-22, while for zlib and lzma, the options are 0-9. `borg create --compression lzma,4 repo/path::archive-name data-sources` creates an archive using the lzma algorithm.

You can also direct *Borg* to decide for itself whether the data needs to be compressed or not. This is done with the `auto` command option. *Borg* attempts to use the lz4 compression on each file, and should the file be compressive, uses the defined compression algorithm and level:

```
$ borg create --compression auto,lzma,4 repo/path::archive-name
data-sources
```

Borg will first test each file's compressibility using the lz4 algorithm. If the file can be compressed, *Borg* will compress it using the lzma with level 4.

all the data you wish to backup. You can place data from different directories into a repository. *Borg* is able to identify files even if the names or location changes. For instance, if file A in location X is already in the repository, when you attempt to backup file B from location Y *Borg* uses chunking to determine if file B is indeed a new file.

Borg has various sub-commands such as `create`, `init` and so on, and each has its own set of arguments. Thankfully, you can access help using the `borg command --help` command, or from the website. To create a repository, you must run the `borg init PATH` command. The project recommends that you use encryption to safeguard your data. Use the `-e none` option if you don't wish to use encryption. Alternatively, you can choose from between using a passphrase only, or passphrase and key encryption.

The `borg init -e passkey /path/to/repository` command will prompt you to enter a passphrase. The repository created with the `init` command will be protected by the passphrase. You will not be able to access the repository, add backup files to it, or restore data from it without the passphrase.

Learn to live with the Borg

Once a repository is created, *Borg* is now ready to accept files. Each backup is stored in a unique archive within the repository. You can create many repositories and archives with *Borg*. For instance, you could create a repository to backup PDF files from work, and another repository to backup music sheets. You must use the `borg create /path/to/repo:archive_name /path/to/data` command to create a new archive within the repository:

```
$ borg create ~/Documents/temp/repo::first-archive ~/Documents/LXF-submissions ~/Pictures/for-LXF/
```

In this command, we've created a new archive called **first-archive**. *Borg* will recursively traverse through the specified directories and add files to the specified archive. When we create a follow-up archive, only new content from these directories is added to the archive:

```
$ borg create --stats ~/Documents/temp/repo::second-backup ~/Documents/LXF-submissions ~/Pictures/for-LXF/
Enter passphrase for key /home/linuxlala/Documents/temp/repo:
Creating archive at "/home/linuxlala/Documents/temp/repo::second-backup"
------------------------------------------------------------------------------
Archive name: fifth-backup
Archive fingerprint: c4ce413d82e4d00b9b4defdf8eaddc8af674059a8044aac905e00c527c8224d8
Time (start): Mon, 2021-09-27 18:33:14
Time (end):   Mon, 2021-09-27 18:33:14
Duration: 0.19 seconds
Number of files: 10
Utilization of max. archive size: 0%
------------------------------------------------------------------------------
                       Original      Compressed    Deduplicated
This archive:          35.43 MB        33.91 MB        2.39 kB
All archives:         140.38 MB       134.28 MB       33.80 MB
                Unique chunks         Total chunks
Chunk index:               38                  115
------------------------------------------------------------------------------
```

Refer to the last column in the Deduplicated size. This tells you that the archive only has 2.39K of new data, as all the other data in the specified directories is already part of another archive.

Mount and extract archives

As its name suggests, you use the `borg extract` command to extract data from an archive. By default, *Borg* will extract the entire content of the specified archive, but you can choose the data you wish to extract using the `--exclude` option, or even the PATH.

Using our previous example where we created an archive of two directories **~/Documents/LXF-submissions** and **~/Pictures/for-LXF/**, we can extract all the image files using either pattern matching or the directory path:

```
$ borg extract ~/Documents/temp/repo::first-backup ~/Pictures/for-LXF
Enter passphrase for key /home/linuxlala/Documents/temp/repo:
```

You can similarly use the `--exclude` command option to restrict *Borg*:

```
$ borg extract ~/Documents/temp/repo::fifth-backup –exclude '*.odt'
Enter passphrase for key /home/linuxlala/Documents/temp/repo:
```

The `borg mount` command can be used to mount an archive. First create a mount point for the archive:

```
$ mkdir /tmp/backup
$ borg mount ~/Documents/temp/repo::second-backup /tmp/backup
```

You can now open the **/tmp/backup** directory in your favourite file manager and look through the contents. Be sure to unmount the archive with the `borg umount /tmp/backup` command after you're done.

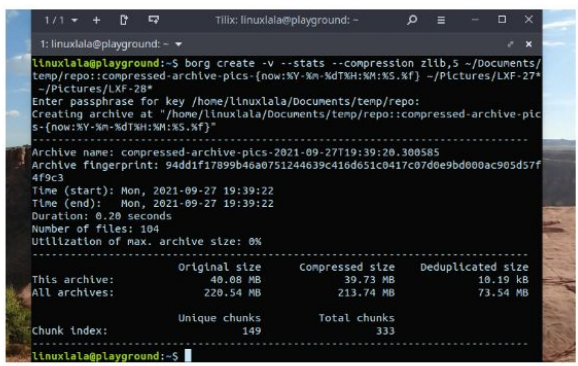
You'll also appreciate the `borg diff` sub-command, which can be used to compare the contents of two archives. The `borg diff ~/Documents/temp/repo::second-backup third-backup` command will inform you of all the changes between archives.

There's plenty more you can do with *Borg*. Thankfully it's an excellently documented utility and you'll find all the information you need in help pages for each of the sub-commands, or on the project's website, to truly understand the full capabilities of *Borg*. LXF

QUICK TIP

The main website of the project is https://borgbackup.org. You can access the official documentation at https://borgbackup.readthedocs.io. The main website also provides links to free resources as well as commercial support and services around Borg.

When creating an archive, you can timestamp the archive for smarter filenames. This is more efficient than numbering your backups manually.



SHARIK

Credit: https://github.com/marchellodev/sharik

Network-share files quickly and easily

Nick Peers reveals how to share files over your local network without getting bogged down in networking protocols.

OUR EXPERT

Nick Peers is reluctantly accepting the mantle of 'veteran', given his relationship with computers stretches back nearly four decades. You may not, however, call him an 'old git'. Not just yet.

Sharing files between two devices on the same network should be straightforward, right? However, anyone who's wanted to quickly shunt a file to or from their mobile device, but struggled getting shared folders set up that all the machines on their network can see or access, might have something to say about that.

You could try setting up a pair of synced folders using a tool like the wonderful *Syncthing*, or make use of cloud storage such as Google Drive or Dropbox, but these all come with their own pitfalls. Synced folders can result in data loss if used incorrectly, while cloud storage has question marks over its security.

This is where a tool like *Sharik* (**https://github.com/marchellodev/sharik**) comes in. It's designed to make it easy to send files to any desktop or mobile device connected to your local network – whether wirelessly or via Ethernet cable – without the need for an internet connection. At the present time, there's a *Sharik* app available for Linux, Windows, Android and iOS for sending files, but you can share files with other systems via their web browser, as you'll see below.

Get Shariking

The quickest and easiest way to install *Sharik* is through snap – it's integrated into the *Software Store* in Ubuntu 16.04 or later, or you can install it from the command line, as follows:

```
$ sudo snap install sharik-app
```

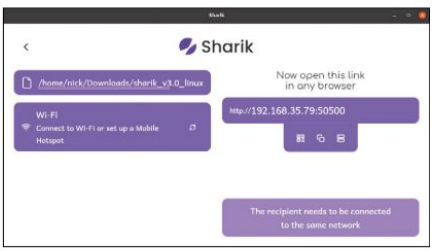
At time of writing, the latest version for Ubuntu 20.04 on snap was 2.5. If you want the very latest version – 3.0 – then you can install it through its own repo instead (skip the first command if *curl* is already installed on your system):

```
$ sudo snap install curl
$ curl -sS https://ppa.mark.vin/key.gpg | sudo apt-key add -
$ echo "deb https://ppa.mark.vin/sharik stable main" | sudo tee /etc/apt/sources.list.d/sharik.list
$ sudo apt-get update && sudo apt-get install sharik
```

For occasional use, there's also a portable AppImage available for direct download from the site. After downloading, don't forget to right-click the AppImage file, select Properties>Permissions tab and tick 'Allow executing file as program'.

QUICK TIP

If you want to quickly share a file with a mobile device outside your network, Sharik will work with your phone's mobile hotspot. Set up the hotspot, connect your other device to that network and you should be able to send and receive files between them.



Once shared, all the details you need to communicate to the receiving device are displayed in Sharik's program window.

The simplest way to use *Sharik* is purely to send individual files to other devices, regardless of whether or not they're running *Sharik*. The receiver doesn't need *Sharik* installed because the file can be accessed through their web browser, as you'll see shortly.

Although *Sharik's* documentation seems to hint that you need to be connected to the same Wi-Fi network to share files, the reality is that any machine on your local network with the same subnet (typically 192.168.0.x, where 'x' is unique to each connected device) should be able to access files sent via *Sharik*.

Let's send our first file. Launch *Sharik* on the sending device – if you're using the desktop version, click Send file to select a file to share, or click Text to send a text message to another *Sharik* user (perhaps to alert them of an incoming file). Use the file picker to select the file that you wish to send.

Once it's been selected, you'll be shown a web link that points to your source device's IP address (for example, **http://192.168.0.2:50500**). If you type this URL into any browser on any device connected to your local network, you'll either be shown the text message you typed, or prompted to save the file you've sent from the source machine.

Lazy types – or people in other rooms – may need an additional helping hand. Beneath the IP address you'll see three buttons: click the left-hand one to reveal a QR code that you can scan into your phone or tablet for quick-fire access in its default browser. The middle button simply copies the URL to the clipboard for pasting elsewhere – for example, into a chat message.

Press the right-hand button to switch between network interfaces. It's only relevant if you're connected to two networks via different interfaces (say Wi-Fi and Ethernet), so most people won't require this option.

Note that the file being shared remains available to any – and all – locally connected devices until you either close *Sharik* or click the back button to select another file to send. When you click back, you'll notice that *Sharik* remembers files that you've previously shared via a handy History list, so you can easily re-share files without having to locate them again. There's no apparent limit to the History list, but should you want to clean it up for privacy purposes, simply click the wastepaper bin icon next to History to remove them all and restore a blank slate.

Receive files with Sharik

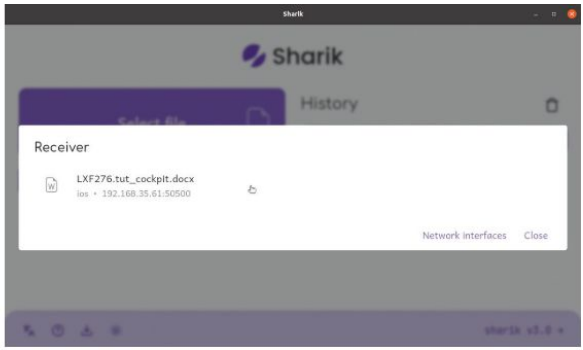
If both your devices are on the same network and running the *Sharik* app – which basically covers everything but macOS at the present time – then you can speed things up a little by putting the destination device into Receiving mode. Launch *Sharik* and you'll see four buttons in the bottom left-hand corner of the program window. Click the Download button and you'll see a Receiver pop-up window appear. In the rare event you have two network interfaces configured and connected to different networks, click Network interfaces to switch between them. You should be able to identify them by IP address and WLP... (Wi-Fi) or ENP... (Ethernet) identifier.

Now switch to the device containing the file you wish to transfer. Open *Sharik* and select the file in the usual way; once it's been selected, you should see the 'Receiver' pop-up window change to display the file being offered. Click this and it'll download the file via your browser.

If you send a file from a machine with two network interfaces, then two instances of that file will appear in any *Sharik* instance in recipient mode, one for each interface's IP address. Both links should work, but it's worth noting so you know that it's not a glitch.

Internet sharing

Sharik's major limitation is that it only works over your local network. If you'd like to share files over the internet with friends, family and colleagues, you'll need to try a different approach. This comes in the form of *ShareDrop*, an open-source, web-based P2P tool.



Sharik can also be used in receiving mode, which basically provides a convenient shortcut to the file being shared.

ShareDrop works over both your local network and the wider internet. When planning to share over your local network, open **www.sharedrop.io** in your browser on both devices and you should see your local computer appear at the bottom of the screen with a name like 'Practical Cat'. After a short pause the other device will appear above it, again with a name like 'Warmhearted Rhino'. On your source computer, either drag a file on to the destination icon, or click the icon to open a File Upload dialog.

Once done, you'll be prompted to confirm your choice: click Send to do so. The recipient should then see a similar pop-up message appear asking them to decline or accept the file. Accepting it results in the file being transferred, and then the recipient will be asked where to save it.

When sharing over the internet, you'll need to either click the + button in the top right to generate a link that you can share with a remote user, or scan the QR code into your mobile device. *ShareDrop* uses WebRTC in your browser to generate a direct – and encrypted – P2P link between sender and receiver, so the file goes directly between them without passing through any intermediary servers, ensuring your security – and privacy – are protected. LXF

» MOBILE SHARING OPTIONS

You'll find additional sharing options are available when using *Sharik* from your mobile. For example, Android users can share installed apps between devices – these are transferred as .apk files. Apple users can share photos using the Gallery option (Android users can share their photos via the Files option).

Although a version of *Sharik* is available from the *Google Play Store*, this is outdated. A better bet is to install the latest version – 3.0 at time of writing – from the open-source *F-Droid* (visit **www.f-droid.org** to get started if you haven't already got it).

Once you're sporting the latest 3.0 release, you'll be able to share content directly from apps using the system Share button – in Android, after tapping Share, select More under 'Share to apps' and you should see *Sharik* appear as an option.

In an ideal world, *Sharik* will behave impeccably every single time you use it. In reality, however, that's not always the case. We had no problems when sending and receiving files between Linux machines and Android devices. However, we were unable to receive anything sent as a 'file' from iOS. The workaround is to send text, photos or share content via the system sharing tool within apps, which works flawlessly.

This Android screen was transferred to Linux in Sharik for this tutorial. Nice and simple.

BACK ISSUES » MISSED ONE?

ISSUE 281
October 2021

Product code:
LXFDB0281

In the magazine
Discover how to customise the Mint 20 desktop with our in-depth feature. We explore virtual private servers, assess five open source art programs, build a Pi-powered pinhole camera, recreate pseudo-3D racing games and emulate the Atari 800.

DVD highlights
Mint 20.2 "Uma" and Elementary OS 6.0 "Odin" (both 64-bit).

ISSUE 280
September 2021

Product code:
LXFDB0280

In the magazine
As Linux hits 30 years, we show how the kernel became a worldwide phenomenon. We take a look at terminal browsers, plus show how you can make your mark in desktop publishing, build a Pi Nextcloud server and run Linux distros from yesteryear!

DVD highlights
Pop!_OS 21.04 (64-bit), and classic Linux distros (32-bit, unsurprisingly).

ISSUE 279
Summer 2021

Product code:
LXFDB0279

In the magazine
Find out how to use your Raspberry Pi to stream video to the world. Elsewhere, we compare five office suites, diagnose and solve Linux problems, emulate the Acorn Electron, set up a virtual network, design circuit boards and manipulate date with Pandas.

DVD highlights
Rescue kit (CloneZilla, System Rescue and Rescuezilla, plus Zorin OS Lite.

ISSUE 278
August 2021

Product code:
LXFDB0278

In the magazine
Want a faster, better server? We show you how to set up some of the best server distros around. We also cover desktop virtualisation, running a mobile second screen, emulating the Altair 8800, multitasking in Python and web-app security.

DVD highlights
Bodhi 6.0 and Lubuntu 21.04 (both 64-bit) and AntiX 19.4 (32-bit).

ISSUE 277
July 2021

Product code:
LXFDB0277

In the magazine
Discover what's new in the latest version of Ubuntu, grab a slice of network-attached storage, code a game in Scratch, emulate the Dragon 32 and set up your own streaming server with *Jellyfin*, Plus we look back at Prestel, the pre-internet data service!

DVD highlights
Ubuntu 21.04 (64-bit) and MX Linux 19.4 (32-bit).

ISSUE 276
June 2021

Product code:
LXFDB0276

In the magazine
We show you how to set up your own server for your photos, either at home or in the cloud. Learn how to emulate the Commodore PET, manage headless servers with *Cockpit* and render objects in *Blender*. Plus get coding and graph metrics in Python today!

DVD highlights
Manjaro 21 (64-bit) and Tails 4.17 (32-bit).

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

EMULATION

Credit: http://48k.ca/trs80gp.html

Running the classic Tandy TRS-80

Les Pounder travels back in time to when Doctor Who had a mop of curly hair and a trio of computers ruled our homes.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker. He has worked with the Raspberry Pi Foundation and he blogs about hacks and makes at bigl.es.

Looking back, 1977 was a pretty good year. We had the space shuttle Enterprise being put through its paces, a plucky farm boy single-handedly destroyed a moon-shaped space station and learnt about space wizards, and we had a trio of excellent home computers.

We've already covered the first two machines in that trio – the Commodore PET and Apple II – in previous issues of *Linux Format*, but the final machine, the Tandy TRS-80 isn't a machine that most UK readers will have had the chance to use. Nonetheless the Tandy TRS-80 was a big deal in its native US market. It was released on 3 August 1977 and it was sold via Radio Shack (TRS actually refers to Tandy Radio Shack) for approximately $399. In 2021 the price would be a hefty $1,799 adjusted for inflation! For your money you got a Zilog Z80 CPU running at 1.774MHz, 4KB of RAM and a comfortable QWERTY keyboard. A bundled cassette drive was your gateway to a wondrous world of eight-bit gaming.

The base system was enough for home users and hobbyists to cut their teeth with the burgeoning home computer scene, but Tandy also had a series of upgrades to keep your TRS-80 updated and to add to its annual profits. You could purchase aftermarket RAM upgrades, up to a whopping 48KB (in 16KB packages) and replace the slow and fiddly cassette drive with floppy drives and even a hard drive.

In 1982 the TRS-80 line was the best-selling home computer, eclipsing the popular Apple II that saw only a fifth of the TRS-80 sales. The TRS-80 was later renamed to the TRS-80 Model 1, and we saw the TRS-80 Model 2 released in October 1979 – a machine that broke compatibility with the original TRS-80. The Model 2 was aimed at small businesses and came with a 4MHz Z80, 32KB RAM and an eight-inch floppy disk drive. Software for the Model 1 and 3 was unfortunately not compatible with the Model 2, largely because of differing disk formats and system architectures.

CREDIT: Dave Jones, EEVBlog CC BY 2.0, www.flickr.com/photos/

The silver case and flat black finish indicates the time from which the TRS-80 was born. Classic looks for a classic era of computing.

Without further ado, let's step into our time machine and travel back to the year 1977, and experience the wonders of the TRS-80 via the magic of emulation.

Tandy time

Our emulator of choice for this issue is *trs80gp*, an emulator for all models of TRS-80 and one that greatly simplifies the process of emulating the machines.

Installation is as simple as downloading the file from **http://48k.ca/trs80gp.html** and extracting the contents. Inside the archive are versions for Windows, Mac and Linux. There are two Linux directories: one for 64-bit and the other for 32 bit. The file inside the directories is already set to executable, so a double-click is all you need to do to start the emulator.

The emulator starts as a TRS-80 Model 3, and this machine has compatibility with the Model 1 so all is good. One difference though is that the Model 3 in this emulator starts an operating system called TRSDOS, and so to reach the BASIC interpreter we need to type

QUICK TIP

The INPUT keyword can only be used in a program and not directly at the interpreter. We found this out after far too much Googling. If you want to use the INPUT keyword, or test out an idea, make sure that the line starts with a number (100,200, etc).

BASIC and press ENTER We're then asked "How many files?" and "Memory". We can dismiss both of these by pressing ENTER.

Tandy BASIC

Let's flex some of our BASIC muscles. We start as always with the ol' 10 PRINT project. Each line of BASIC code for a project will start with a number: 10, 20, 30 etc. This tells the interpreter the sequence of code; it jumps from one line to the next in ascending order. But why do we do this? Quite simply, if we make a mistake and miss out a line of code we can insert another line of CODE without messing up the original code. Let's do the 10 PRINT project to illustrate this:

```
10 PRINT "HELLO WORLD"
20 GOTO 10
```

If we RUN this code it'll print `HELLO WORLD` over and over again. Press F7 on the keyboard to break the running code. But what if we want to add another line? We can insert a new line between 10 and 20. Logically this would be 11, giving us many more options to expand or correct the code. But we are going to use 15 because this is just a simple test:

```
10 PRINT "HELLO WORLD"
15 PRINT "LXF ROOLZ"
20 GOTO 10
```

Now RUN this new code and you will see alternating lines of `HELLO WORLD` and `LXF ROOLZ` on the screen. Press F7 to stop the code.

Let's write something a little more challenging. There's a popular game in the US, *Mad Libs*, which creates stories that feature words given at random by other players. The goal of *Mad Libs* is to teach common written language skills and rules. But most of the time it descends into making up silly sentences. The randomly chosen words are often verbs and nouns, but often players are asked for places and celebrity names. Our version is greatly simplified for this tutorial, enabling us to demonstrate how to create string variables in BASIC, and then drop them into a sentence.

We start with line 10, and this is a simple print statement, giving an instruction to the player.

```
10 PRINT "TYPE IN THREE WORDS TO CREATE A MESSAGE"
```

Line 20 is another instruction, asking for a place:

```
20 PRINT "A PLACE"
```

Line 30 used INPUT to capture what the user types and store it as a string variable, **A$**. In other versions of BASIC we can combine lines 20 and 30 to create a prompt that asks the question and stores the answer, but BASIC on the TRS-80 won't let us do that.

```
30 INPUT A$
```

```
>LIST
10 PRINT "TYPE IN THREE WORDS TO CREATE A MESSAGE"
20 PRINT "A PLACE"
30 INPUT A$
40 PRINT "A VERB"
50 INPUT B$
60 PRINT "A CELEBRITY"
70 INPUT C$
80 PRINT "MY FAVOURITE FOOD IS "A$" AND I LOVE TO EAT IT EVERY "
B$" WITH MY BEST "C$
READY
>
```

The BASIC code for our project is simple on purpose. We want to show how easy it is to work with user input, storing it into a variable and then using the variables to replace string sections.

Line 40 is another question to the user. This time it asks for a verb, and the answer is captured to a string variable, **B$** using the code on Line 50:

```
40 PRINT "A VERB"
50 INPUT B$
```

Line 60 repeats the process, asking a question and line 70 stores the answer as string variable **C$**.

```
60 PRINT "A CELEBRITY"
70 INPUT C$
```

The final line, 80, is where the sentence is constructed. We use a print statement to write the sentence to the screen. We have hard-coded phrases, contained in quotation marks, in between each phrase we have a string variable **A$**, **B$** and **C$**, which replaces the variable name with its contents. Keep an eye out for spaces between the phrases and variables, otherwise you'll produce mashed-up words.

```
80 PRINT "MY FAVOURITE FOR IS "A$" AND I LOVE TO EAT IT EVERY "B$" WITH MY BEST "C$
```

With the code ready to go type RUN and press ENTER to start. Now enter the place, verb and celebrity and have a laugh or two.

Tandy gaming

The easiest way to play a TRS-80 game is via JavaScript. Each issue we feature another JavaScript emulator for a retro machine and we never tire of seeing this excellent language being used outside of the web.

QUICK TIP

Want to try out a different TRS-80 machine? You can in trs80gp. Go to File>Machine and take your pick. Each of the machines will boot into TRSDOS, awaiting your command.

» THE TRS-80 RANGE

There were many models of TRS-80. Some were compatible with each other, such as the Model 1 and 3. The later model 4 was backwards compatible with software written for the 3, but that was it. The TRS-80 Model 4 was a faster and better-specced machine with integrated disk drives. The Model 4P was a portable version of the Model 4, and by portable we mean "luggable". For true portability we have to look to the TRS-80 Model 100.

The TRS-80 Model 100 wasn't a "true" TRS-80. Rather, it was a rebadged Kyotronic 85, so software compatibility between the machines was broken. The Model 100 was the tablet of the 1980s. It had a comfortable keyboard, decent LCD screen and it ran for up to 20 hours on AA batteries. It was popular with journalists of the era, largely for the keyboard, but also because it could send electronic messages via a conventional phone line via a modem. So a journalist could write their copy at an event, and send it to the newsdesk for publication.

If you wanted an even smaller TRS-80 then Tandy, then the TRS-80 Pocket Computer PC-1, PC-2 and PC-3 gave us an ultra-small form factor computer for basic tasks. The models could be attached to external devices such as printers and micro cassette drives to provide a typical computer experience.

The first JavaScript emulator is at **http://trsjs.48k.ca** and it comes with a limited number of games, including *Time Trek*, an unlicenced *Star Trek* clone where the Enterprise battles Klingons. More often than not the Klingons won our battles. Another game on the list is *Sea Dragon*, a side-scrolling submarine game where we dodge mines. The action is a little slow paced, although not as slow as *Das Boot*, but take the difficulty up to expert and your skills will be put to the test.

Emulating retro machines is not just for JavaScript. We found *JS Mocha*, an HTML5 and JavaScript emulator which uses HTML5 for local file storage. *JS Mocha* emulates the Tandy CoCo 2 machine, which is much newer than the Model 1. And via the user interface we have a plethora of cassette- and disk-based games to play. Head over to **www.haplessgenius.com/mocha** and have a play one rainy afternoon.

To play a game with our emulator is possible, but we first need to find games. If you have these already then no problem. If not then we leave it as an exercise for your Google skills. We can load cassette or floppy disk images into the emulator. For cassette images use the Cassette>Insert... menu and then navigate to the desired file. If you're in TRSDOS then you'll need to type `BASIC` to open the BASIC interpreter, and then use either `CLOAD` or `CLOADM` to load the tape.

In our tests the tape playback was non-functional, and by that we mean it caused our Ryzen 5 5600X machine to get rather warm as it tried to play a tape. Floppy disks, on the other hand, had no issues at all. To play a floppy disk, go to Diskette>:0 and Eject the current disk. Then go back to that menu and use Insert Disk, navigate to your disk image (DSK file) and select the file. Now go to File>Reset Button and reboot the TRS-80. This will auto-load the floppy disk. Now we can play *Sea Dragon* and try to dodge those mines!

The BASIC interpreter was a key part of many home computers of this era. From here we can launch our games, write new code and learn skills that will last a lifetime.

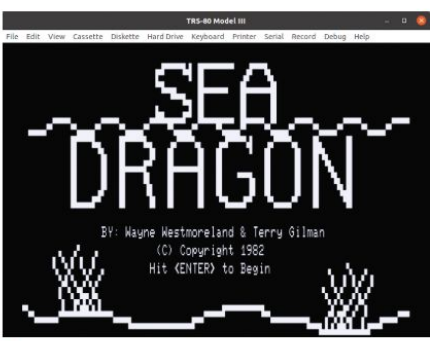
Explore the demoscene

The 1970s saw computers come into the home, and with it more people had access to learn new skills. While getting to grips with the technology, artists and coders would learn new skills and the demoscene is where we see the output of their work.

This early on in the history of home computing, the demoscene was just forming and so there are slim pickings for the Model 1. We need to move forward to the 1980s and the CoCo in order to see more technical demos. But one did catch our eye, and it just happens to be one of the most popular demos for the Model 1. The Dancing Demon, created in 1978/79, sees exceptionally fluid movement timed to the beat of a beep. The demon moves with grace and we can see hands and antenna reacting to the beat of the music. Clever work considering the limitation of the machine. The Dancing Demon and other demos are viewable via YouTube at **https://youtu.be/-mAiDntWDfM**.

The TRS-80 in 2021

UK readers, we hope you have lots of money as it seems that online prices of TRS-80 are geared more towards America, the home of the TRS-80. We spotted "buy it now" units for a few hundred pounds, and accessories can be found for not much cheaper. It's the later models, the Tandy CoCo (Color Computer) where we see more choice and lower prices. If you read our Dragon retro tutorial (**LXF277**) then you'll know that the UK had a Welsh "Tandy CoCo" compatible. No matter



Do you want to be a submarine commander? Have you watched The Hunt for Red October over five times? Then Sea Dragon is for you!

» RESOURCES USED IN THIS FEATURE

Writing this feature involves a lot of resources and we wanted to share them with you, so that you can get up and running with your emulator and the real hardware.

TRS-80.com has lots of resources for beginners and old hacks. There we learnt about the BASIC syntax for this machine, and helpfully the site has a reference page which breaks it down into common programming topics. See **www.trs-80.com/wordpress/info-level-2-basic-language**.

The emulator used in this tutorial was *trs80gp* and its website at **http://48k.ca/trs80gp.html** is a gold mine of information. We made extensive use of it when learning to use the emulator.

TRSDOS is a little different to what we are typically used to and so we needed the TRSDOS and Disk BASIC reference manual in order to steer us true. Find it at **https://archive.org/details/TRSDOS_v2.1_Reference_Manual_1979_Tandy/mode/2up**.

Another TRS-80-specific BASIC guide and this was a lifesaver while we wrestled with the INPUT keyword issue. Read it at **https://bit.ly/lxf-282-trs80-basic-guide**.

A simple and clear resource, the TRS-80 Model 3 Disk BASIC Manual provided clarity when we (often) went wrong. Find your answers at **https://bit.ly/lxf-282-trs80-model-3-disk-basic-manual**.

Finally, many thanks to these Twitter users for their help in diagnosing BASIC programming errors that were made in the course of this feature: **@ErikPetrich, @isleofmandan, @anne_engineer, @ExtElec, @ukscone, @0possum, @ZXSpectROM, @mfraz74, @dwculp** and **@ForrisHilier**.

This black and white image does little to sell how fluid the animation is for a late 1970s computer. An exceptional piece of coding for the era.

what you buy, you need to make sure of a few key things. First of all, if buying from the US, note that US mains outlets run at 110V, not the 240V common in the UK, so you'll need to purchase adapters/power boards to run using 240V. Don't just plug it in and pray, otherwise you'll release the blue magic smoke.

Next, when buying any used tech, make sure you understand what work is involved. A quick wipe down and Retr0bright could be your only task, or a complete re-cap (replacing all the capacitors) might be your weekend task. Lastly, contact user groups for your machine. Be part of a network of experts who can guide you based on their experiences. Before attempting any renovation work, make sure you know what to do. If in doubt ask someone! A badly renovated machine could be dangerous, or ruin a scarce piece of history.

If you already have a TRS-80 then what can you buy? As usual the community provides ample accessories. We can purchase SD card-based floppy drive emulators that connect to real TRS hardware. From **https://hxc2001.com** we can purchase the hardware and software to emulate a floppy drive, with an SD card storing thousands of applications, games and utilities. All we need to do is select the disk image from the SD card, load it up and then run. Much more reliable and easier to obtain than eight-inch floppy disks.

CoCo users can pick up the CoCo SDC, which connects to the expansion port. We also spotted fast tape emulators that claim to load tapes 50 times faster than the real thing. These drives cost approximately £70 via online auction sites. Model 3 and 4 owners can pick up a modern day hard drive emulator for $210.

The legacy of the TRS-80

The TRS-80 was the machine that launched many developers careers. It was a low-cost, popular and easy-to-use machine and it had a solid catalogue of games and utilities. The add-ons provided by Tandy and third parties kept the machine alive, along with the many iterations that were released over the years, including a portable mode. It may not have had much impact in Europe, but in the US this was the machine to cut your programming teeth on. LXF

QUICK TIP

There are two commands to load a cassette game. CLOAD is used to load games written in BASIC, whereas CLOADM is used to load games written in machine code.

The trs80gp emulator is the simplest way to recreate TRS-80 series machines. It's ready to go and has a simple user interface to load cassettes and disks.

SERVERS

Part One! Don't miss next issue, subscribe on page 26!

How to build the Linux Format server

Start running your own server with the help of **David Rutland** and our new series on what to do with one once you've got it!

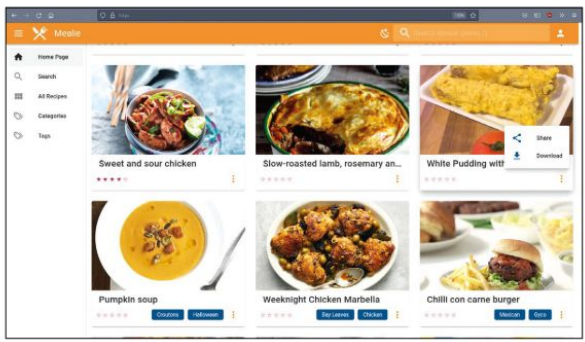
OUR EXPERT

David Rutland is a tinkerer and a dilettante. He buys domains on a whim and runs them from a Raspberry Pi behind the couch.

Okay, so you've decided you need a Virtual Private Server in your life. It's a great idea, and will allow you to run a whole bunch of self-hosted, web-facing software without the noise, expense and energy bill associated with setting up a server in the cupboard under the stairs at home. The magical orphans you already have stashed there will be pleased.

If you're a regular reader of these hallowed pages, you've probably already seen our VPS feature in **LXF281**, which covered such essentials as what a VPS actually is, why you'd want one, how to choose a provider and initial setup. In case you missed it, we've been kind enough to provide a PDF copy of the article which you can find at **https://bit.ly/lxf281lxfserver**. Go and read it then come back.

Up to speed? Good. But there is so much more you need to know, and choices you need to make. This article will help to guide you through some of the basics you need to keep your spanky new VPS safe, secure and working as it should. Don't worry, they're not difficult choices, but they will have an effect on how you install and use software.



This excellent web-facing recipe manager can be up and running with a single command thanks to Docker.

QUICK TIP

Running software services on a Virtual Private Server is as easy or as hard as you want it to be. Deploying a raft of Docker containers is simple, but you'll miss out on what's going on under the bonnet. Try doing things the long way where you can – it will give you a better idea of how your services work.

Actual server software

Your VPS is a virtual machine running in a datacentre in a different part of the world. You're going to be accessing the services you run on it via one or more domain names. But how do you ensure that you connect to the right service?

For instance, if you've installed *Jellyfin* to manage your media, *Mealie* to keep track of your recipes, *Wger* to track your personal fitness and *FreshRSS* to aggregate the latest news on your behalf, you'll need four separate domain names, or at the very least sub-domains (many self-hosted sites need to function from a domain root). And this collection of names will need to point at the static IP address of your VPS.

With all of these incoming requests, your VPS needs to know how to serve the right content. It would be less than useful if, while trying to plan a balanced meal for your upcoming marathon, you were instead served up the latest depressing news titbits from *The Economist*.

Thankfully there's software out there which can help your server to direct traffic to the right directory or port, and return a response that is both coherent and appropriate to what you are trying to achieve.

If you followed part one of our VPS series you will have already installed Apache Server. Apache is old-school cool, and straight out of the box, it can do exactly what we want it to.

Apache employs user-created **.conf** files to tell it what it's supposed to be doing, and in their most basic form they are pretty simple to understand. They contain the name of the virtual server, what ports to listen on and the Document Root location, which is where the files you want to serve up are kept.

Apache conf files should be created in **/etc/apache2/sites-available/**. For instance, the **.conf** file which relates to our fabulous *Linux Format* URL shortening engine (as it was first set up) looks like this:

```
<VirtualHost *:80>
ServerName lxf.by
DocumentRoot /var/www/lxf/public/
</VirtualHost>
```

That's easy to understand. Apache is listening on

port 80 for HTTP requests to **lxf.by**. When any requests to **lxf.by** come in, they are directed to the contents of the directory **/var/www/lxf/public/**.

Apache **.conf** files will, later on, contain more interesting details such as SSL certificate locations, internal port mapping and access controls. You don't need to worry about those right this second.

With Apache, for each service you use you will need to create a **.conf** file providing the relevant details; activate it by typing

```
sudo a2ensite /etc/apache2/sites-available/
yournewservice.conf
```

and then restart apache with:

```
sudo service apache2 restart.
```

When directing traffic to individual domains, Apache is acting as a reverse proxy. But Apache isn't the only server software in town which can manage your traffic for you. There are dozens of projects which are more or less specialised, easier to use or more comprehensive.

Apache's chief competitor is Nginx, which, like Apache, serves around one quarter of the entire traffic on the internet. It's faster and more lightweight than Apache, but it's more difficult to install and configure, and more importantly, doesn't allow admins to override system-wide access settings on a per-file basis.

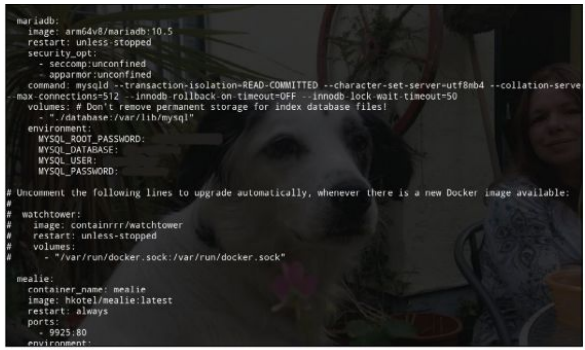
Nginx is also embroiled in an ongoing ownership dispute in the Russian Federation. We don't feel comfortable recommending a piece of software – regardless how good – which could be yoinked from the world at any moment.

tOther server software capable of running as a reverse proxy includes Hiawatha, WEBrick and Cherokee. Although each of these have their own benefits and advantages, they are not nearly as popular as Apache or Nginx, so it will be considerably more difficult to find online support from fellow users.

Security matters

See that little padlock up in the corner of your address bar? Sometimes, it will say secure, or TLS, or SSL. It means that the traffic between your machine and the webpage you're connected to is encrypted.

In theory, no-one can eavesdrop on exactly what you're doing, and even your ISP can only see what site you're visiting – not what content you view while you're there. It also means that the site is exactly what the address bar says it is. The content at **https://linuxformat.com** is created by us here at *Linux Format* – there's no man-in-the-middle injecting malware into the HTML headers or siphoning off the credit card details of our users. In addition to the tiny padlock and three letter abbreviations, you'll note that the address starts with HTTPS rather than HTTP. The S stands for secure.



```
  mariadb:
    image: arm64v8/mariadb:10.5
    restart: unless-stopped
    security_opt:
      - seccomp:unconfined
      - apparmor:unconfined
    command: mysqld --transaction-isolation=READ-COMMITTED --character-set-server=utf8mb4 --collation-serve
--max-connections=512 --innodb-rollback-on-timeout=OFF --innodb-lock-wait-timeout=50
    volumes: # Don't remove permanent storage for index database files!
      - "./database:/var/lib/mysql"
    environment:
      MYSQL_ROOT_PASSWORD:
      MYSQL_DATABASE:
      MYSQL_USER:
      MYSQL_PASSWORD:

# Uncomment the following lines to upgrade automatically, whenever there is a new Docker image available:
#
#  watchtower:
#    image: containrrr/watchtower
#    restart: unless-stopped
#    volumes:
#      - "/var/run/docker.sock:/var/run/docker.sock"

  mealie:
    container_name: mealie
    image: hkotel/mealie:latest
    restart: always
    ports:
      - 9925:80
    environment:
```

The docker-compose.yml file can be used to start and manage dozens of services. The only required skill is a mastery of Ctrl+C and Ctrl+V.

On the internet, there's a chain of trust that begins with a root certificate provider, continues through resellers and ends up with you – the server admin and end user. From the certificate provider downwards, everyone in the chain needs to prove that they are who they say they are, and that a site is not masquerading as another completely different site.

That kind of reassurance is good to have, especially when connecting to your own private content on your own virtual private server. You may not have anything to hide, but you probably don't want anyone else looking at it anyway.

If you've ever bought a hosting package, you may have noticed among the available extras there's the option to pay extra for SSL. Namecheap, for instance, charges oddly specific amounts of between £7.23 per year and £114.86 per year for the privilege of having the little padlock, and the peace of mind which goes along with it.

Thankfully, as owner and admin of your very own Virtual Private Server, you can provision your various sites with as many security certificates as you desire – at zero cost! The certificate provider we'll be using is Let's Encrypt and the tool is called *Certbot*. Like all good Linux programs it does one thing and does it very well.

QUICK TIP

Remember that a Virtual Private Server is just that – virtual. If you mess up to the extent that you need to wipe it and start again, you haven't lost a lot. Treat it as a learning experience.

» A PATCHY SERVER

Last issue we went into depth on selecting and standing up a bare VPS running Ubuntu Server (**https://bit.ly/lxf281lxfserver**).

If you missed it go back and catch up, else on a fresh install get Apache up and running this way. Ssh in as root, eg. `root@XXX.YUR.IPX.HRE` to your new server. Create a new, non-root user with its own home directory skeleton using the *useradd* command:

```
useradd -m <anew_username>
```

followed by a new password for your new user using the *passwd* command:

```
passwd <anew_username>
```

Grant your new user sudo powers so you can actually get things done by typing the following:

```
usermod -a -G sudo <anew_username>
```

After this you should log out as root and log back in as your new user. Update and upgrade the already installed packages using `sudo apt update` and `sudo apt upgrade` commands and start adding packages which will make your VPS journey possible. As a bare minimum, you should have server software such as Apache or Nginx (we're sticking with the former). You'll also need PHP and a database such as MariaDB.

You can install Apache by typing `sudo apt install apache2` , PHP by typing `sudo apt install php` , and MariaDB by typing `sudo apt install mariadb-server` then `sudo mysqlsecureinstallation` and following the prompts. See, super easy!

QUICK TIP

Certbot can be used to create wildcard certificates which will cover any new subdomain you create. This saves time and can improve security through obscurity, but comes with its own set of risks.

First of all you need to add the certbot repository:

```
sudo add-apt-repository ppa:certbot/certbot
```

then `sudo apt update` and then install *certbot* and dependencies by inputting:

```
sudo apt-get install python3-certbot-apache
```

While it's tempting to just:

```
sudo apt install certbot
```

don't do this. Yes, *certbot* will install, but it won't know how to interact with your server, and you'll be embarrassed when you have to re-consult this section to do it properly.

A dead cert

Congrats. *Certbot* is now installed on your system and ready to get to work! For this next part, we're going to assume that have a website on your server with a domain or subdomain tied to it, such as **test.lxfby**, and that you've created and activated an Apache **.conf** file with the correct configuration.

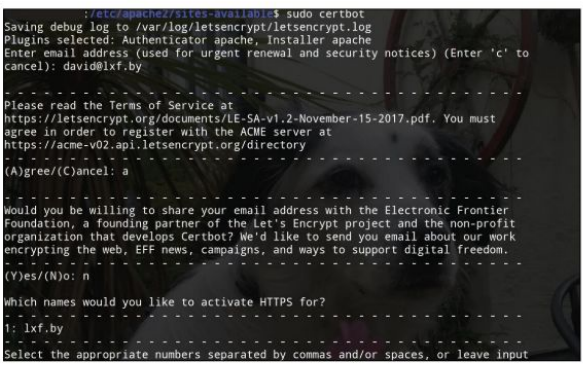
It doesn't matter what the site is, so long as it has at least one page, and is accessible from the internet. The default Apache page will do fine. Running:

```
sudo certbot --apache -d yourspankynewdomain.com
```

results in *certbot* showing you a list of domain names for which you have active **.conf** files. At this point, it's likely that you only have one of them, so it will be a very short list.

Type in the number of the domain name you want a certificate for and hit Enter. After checking that the site actually exists, and it is on the same VPS running *certbot*, the certificate authority Let's Encrypt will issue a certificate and an encryption key which *certbot* will store on your system. *Certbot* will pester you to add your email address so that the Electronic Frontier Foundation can send its monthly newsletter to your inbox, but don't feel pressured to say yes.

Using certbot to get the certificate and keys which make your VPS-based site secure is super-simple.



D is for Database

Setting up a VPS is all about choice, and nowhere is that choice more evident than with the availability of databases you can deploy.

Why do you need a database at all, we hear you ask! The answer is that the services you will be creating on your VPS are going to creating and retrieving vast quantities of data, and they need to store it efficiently.

A web-facing jukebox, as a simple example, would need to know the whereabouts of every track on every album, plus the cover art in three different sizes. It needs to be able to show you the release date, band members, bitrate and duration of the media. It needs to be able to present all of that to you without diving into the ID3 tags for every track every time you refresh the page, which would slow things down massively.

Databases help with this by keeping the data neatly structured and instantly accessible to whichever program needs them. If you've not used databases before, the names can be baffling. Who, exactly, is Maria? Does Postgres have something to do with vibrant modern art? Wasn't Mongo the standby drummer in a knock-off Beatles cover band? No-one knows, or more accurately, cares.

Ideally every self-hosted service you install should be able to use your database of choice to safely store its own data, without being able to access or alter any of the data put there by other programs. *Jellyfin*, for instance, shouldn't be able to access details about your *Jitsi* call logs or check out the private photos in your *Photoprism* instance.

Occasionally, the self-hosted software you plan on running will recommend that a particular database be used, and sometimes that will be a hard requirement. Among the most commonly employed are MariaDB/MySQL, Mongo and Postgres. They all work differently, and there are valid reasons for software developers to favour one over another.

Databases fall into one of two categories: relational databases and non-relational databases. MySQL, MariaDB and PostgreSQL fall into the first category ,

» DIFFERENT STROKES

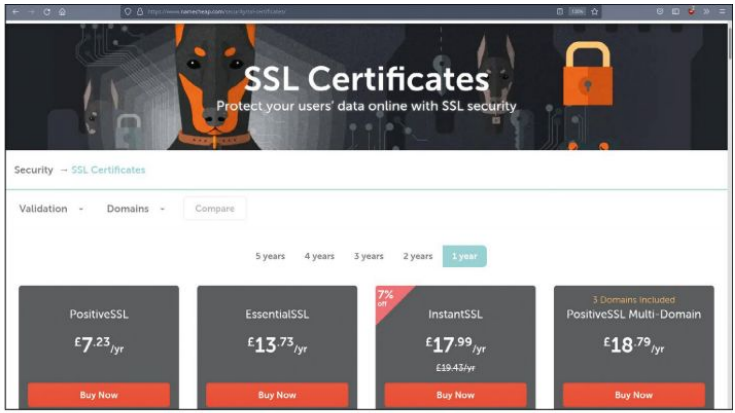
Thousands of people host their own servers, either at home or on a VPS, and their reasons for doing so vary from the need to have everything completely under their own control to trying to push the tech giants out of their life altogether.

Their setups will vary as much as their reasons for having them! We've gone with an Ubuntu server install for the *Linux Format* VPS, purely because it's the most common option, and there is a tonne of support out there for Ubuntu. Likewise, we're using Apache for our proxy server and MariaDB for our main database. You don't need to follow us exactly. You should have fun with your VPS and configure it in the way that makes the most sense to you.

Want to run the latest Arch build? Go for it. Prefer Nginx, and to eschew databases altogether because you have a better idea? That's entirely up to you. It's all about customisation and building your own ideal machine. If you really, really want to, you can even deploy a licensed copy of Windows Server 2019 on your VPS. You shouldn't though.

while MongoDB is in the second. A relational database stores data in tables containing specific pieces and types of data. For example, a photo gallery could store details of pictures filenames and paths in one table and details such as GPS coordinates, camera type, exposure and so on in another. This form of data storage is called structured data.

A non-relational database is different in that it stores its data in a non-tabular form. Instead, non-relational databases tend be based on data structures like documents, which can be highly detailed while containing a range of different types of information in different formats.



Seven per cent off may seem like a great deal for SSL certs from namecheap, but less so when you can use certbot to help you get a free certificate from Let's Encrypt.

On your VPS, you'll probably want one of each type. For applications which require database software, we'll talk you through setting up users and tables as and when we come to it.

Docking around

Of course, life would be a lot easier if developers simply packaged everything you would ever need to use their programs into one neat file or command which would set itself up with a minimum of input from you, and include ready-to-roll preconfigured databases, default logins and all dependencies.

Fortunately, such technology already exists, and in the world of self-hosted software, Docker is where it's at. With a single command, Docker can pull an application, its dependencies, libraries and configuration files from Docker Hub, and have them running in a virtualised container accessible through a virtual port on your Virtual Private Server in minutes. It's that simple.

The Docker application itself is super-easy to install from the default Ubuntu repositories with:

```
sudo apt install docker.io
```

Docker should be run when your VPS starts up. To do this you need to:

```
sudo systemctl start docker
```

and then:

```
sudo systemctl enable docker
```

As an example of how incredibly simple it is to get something up and running with Docker, try typing this into your terminal:

```
docker run -p 9925:80 -v pwd:'/app/data/' hkotel/
mealie:latest
```

All done? Provided you don't have any errors (you shouldn't), you'll find that if you navigate to port 9925 on your VPS (such as **VPS.IP.ADDRESS:9925**), there will be an instance of *Mealie* – a fully self-hosted recipe manager – waiting for you. All set up and ready for the default admin user name and password! You didn't need to set up the database: you didn't need to worry about initial authentication. It's a one-line wonder!

But pulling or running Docker images one at a time can be a chore – especially if your VPS is running multiple services. That's time that could be better spent scouring the internet for more self-hosted projects to make your life both easier and more complicated.

That's where Docker Compose comes in. It enables you to create a text file which will run and manage multiple containers at once. With it, you can set persistent volumes, choose which version of the software you want to run, and get more involved in the configuration of your services.

To install Docker-Compose, first install Docker as above and then:

```
sudo apt install docker-compose
```

Boom! Done

The Docker Compose file is called **docker-compose.yml**, and is written in YAML, which stands for YAML Ain't Markup Language. That's because in the early 2000s, if your project didn't employ counter-intuitive recursive backronyms, it was considered uncool, or possibly even LAME.

To edit your compose file use `nano docker-compose`. You'll notice it's completely empty, and waiting for you to fill it with the details of whatever project takes your fancy. On the project page of *Mealie*, **https://hay-kot.github.io/mealie**, you'll see two sample docker compose files: one using SQLite, and one using Postgres. Make your selection, then copy and paste the contents into **docker-compose.yml**. Save the file (Ctrl+O if you're using *nano*), then from your home directory do:

```
docker-compose pull
```

This will pull the relevant images onto your VPS, and `docker-compose up -d` will start them as containerised applications in the background.

The beauty of **docker-compose.yml** is that more and more services can be appended to it. Fancy trying out the *Photoprism* gallery app? Simply get the contents of the default **docker-compose** file from the project page, and stick it into your own. You can download, launch and manage dozens of applications this way. Hundreds! Thousands (*steady now!–Ed*), even!

Do remember, though, that you only have limited storage space on your server, so don't get *too* carried away until you really know what you're doing. LXF

CUBIC

Credit: https://launchpad.net/cubic

Build your own custom Ubuntu distro

Why settle for what the existing distributions have to offer? **Michael Reed** looks at Cubic, a tool for creating your own custom respin based on Ubuntu.

OUR EXPERT

Michael Reed has been respinning Linux for so long that he spins it like a record, right round, baby. Right round, round, round.

Part of the beauty of Linux is the freedom of deployment that distributions offer, but when installing it for yourself or others you'll want to change things. So, why not make your own version, or 'respin'? That's what a tool called *Cubic* is for, and we're going to look at how you use it to modify the standard Ubuntu installation ISO and bend it to your will in terms of content and aesthetics.

As for the difficulty? If you can install packages from the command line and boot from an ISO in a virtual machine, you should find it easy to get started with *Cubic* as the defaults were always usable in our experience. We'll start with the simplest example of just adding some new packages and rebuilding the ISO. This modified ISO can be used as an installer or as a live desktop environment.

Once that's working, we'll show you how to customise it further by making LXDE the default desktop environment, customising that environment and adding some PPAs so that it really does feel like its your own personal spin on how Linux should look and work.

QUICK TIP

When you modify an Ubuntu distribution that makes use of a live CD environment, using Cubic, you're also modifying the live environment. This means that Cubic is perfect for making a bootable ISO with some extra tools on it. All you need to do is select 'Try Ubuntu' when it starts up.

Install Cubic

Cubic expects to run under Ubuntu or Linux Mint or one of their derivatives. If you are running a different distribution, you can still use *Cubic* and follow this tutorial by running Ubuntu in a VM. Begin by installing from PPA. To do this, locate the *Cubic* page on Launchpad (**https://launchpad.net/cubic**) and follow the instructions by cutting and pasting the needed commands from that page.

`sudo apt-add-repository ppa:cubic-wizard/release` adds the repository to your system. `sudo apt update` updates the system so that it can see contents of the *Cubic* PPA. `sudo apt install --no-install-recommends cubic mn` adds *Cubic* itself. Other than that, the installation should then take care of itself in terms of dependencies.

The next step is to obtain an up-to-date Ubuntu installation ISO to work with. We'll use Ubuntu 21.04, but 20.04 LTS (Long Term Service) is a good choice as well. Launch *Cubic* in the normal way that you launch GUI apps or load it in a terminal window for extra progress information. When running, *Cubic* requires no super-user privileges, unlike some tools of this sort. The first page of the *Cubic* user interface enables you to specify the project directory. *Cubic* doesn't run any tests for free space itself, and you'll need quite a lot of space for the uncompressed ISO. The Ubuntu Desktop installation ISO may weigh in at around 2.7GB, but its actual content is about double that as it's compressed using *Squashfs*. We'd recommend having at least 20GB free before you begin using *Cubic*.

We added the LXDE desktop to a standard Ubuntu installation ISO. We also added a custom default backdrop for all new users.

The decompressing and recompressing of an ISO is rather time-consuming and an SSD works faster than a mechanical hard drive for this task. One way of speeding things up is to leave the project directory intact between different build attempts.

This way, the decompression stage of the process only has to be carried out once, and you keep the changes you've already made. Delete the folder and start again if you want a true fresh start at modifying the ISO.

Having specified the working directory for the project, press Next to proceed to the next page, the Project page. Click the icon next to the filename field and specify the base ISO that you plan to use as your starting point for customisations. Once you've done that, most of the fields on this page will be filled in automatically, but you can safely change things like the release name and the name of the output ISO.

Click next to move on to the Extract page. You don't have to do anything on this page, but be warned that it might be a good time for a quick tea break, as extraction can take a few minutes. Once this is complete, you can open the same directory again in the future. Once you've extracted the files from the ISO, you can quit the application at any time as long as you don't interrupt an operation that is in process, and this means that you don't have to complete the customisation in a single session with the program.

The Terminal page

Without doing anything else, you'll be moved onto the next page, the Terminal page, and this is where we'll be spending a lot of our time. *Cubic* employs some Linux wizardry (a *chroot* terminal) to give you a virtual terminal that operates on the filesystem that will later be rolled into an installation ISO.

In general, most customisations that you can carry out from the command line on a running Linux system can be done from here, and you can use familiar tools, such as *apt*, to carry them out. More advanced customisation can get quite technical, but on the positive side, there is practically no limit to the changes you can make. Note that we can cut and paste commands into the window. There is also a copy icon at the top of this window, and this allows you to copy files and folders into the currently selected directory.

Before we attempt to add packages to the system, we'll start by adding the Universe and Multiverse repositories and update the system.

```
add-apt-repository universe
add-apt-repository multiverse
```

Notice that we omit the `sudo` command as we're effectively the root user already, but show some care as you could accidently wreak havoc on this virtual system as we can affect any files we like within it. If we run `apt upgrade`, the entire system will be updated to the current version of each package. This doesn't increase the size of the eventual ISO by much because you're only replacing outdated packages with newer versions.

GIMP and *Inkscape* are two highly useful graphics programs, and we'll add them both by typing `apt install gimp inkscape`. When you use *apt* in this way, before you confirm that you want to go through with the installation, *apt* will give you an estimate of how much space will be used up; although, it is impossible to say precisely how much size this will add to your finished installation ISO as the filesystem will be compressed. We could have used *Krita* rather than *GIMP* for this example, but we didn't because the *Krita* package pulls in quite a lot of KDE resources, and we're trying to keep this example fairly slim.

Our idea of an appropriate Linux startup screen. Check out the Plymouth themes at www.gnome-look.org or modify/create your own (see https://wiki.ubuntu.com/Plymouth).

What we've done so far is a fairly minor change to the installation ISO, and we'll eventually do things like adding PPA repositories and changing the desktop environment, but for now, we'll leave it at that.

Optimise packages

Clicking Next again takes us to another screen (after *Cubic* has carried out a bit more preparatory work) and this page allows you to remove packages from the final installation. This means that you can have packages that are on the live ISO, but are not part of the actual installation. Make alterations in this section with care, and don't remove anything unless you're absolutely sure that the system doesn't need it.

As is true at all stages of customisation, you can move backwards and forwards through the pages of the *Cubic* interface without losing the virtual filesystem that contains your modifications.

Clicking Next takes you to a page where you can select between three tabs. 'Kernel' allows you to choose

QUICK TIP

There are a lot of Linux utilities for remixing an existing distribution floating about, but we found that most of them aren't maintained! This means that they only work properly with distributions that are now out of date. Even if they seem to work, chances are they'll fall over halfway through the process or what they produce won't work properly.

» CUBIC DOCUMENTATION

The documentation for *Cubic* is rather limited as it's concentrated on the Launchpad page for the project. The two areas that provide the most information are the Answers and FAQs sections. The Answers section shows a good (but not massive) level of activity and it helps that it's searchable too. The main *Cubic* developer is highly active on this forum-like section of the Launchpad page, often offering highly detailed answers. This means that the information is there, but it's quite a lot of work to search for it.

Cubic is quite a well-known program and it's been around for a number of years, so web searches tend to be fruitful. For example, searching on **askubuntu.com** produces a lot of useful information. However, check the age of posts as they stretch back quite a long way and might not be applicable to the version of Ubuntu that you are using. A few YouTube videos covering the basics exist.

We feel that a lack of plentiful, traditional documentation is probably the weakest point of the overall *Cubic* experience.

Asking around on the Launchpad is probably your best bet for answers.

QUICK TIP

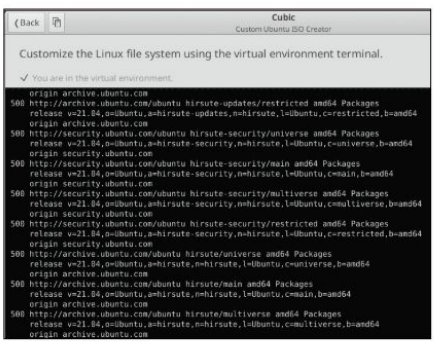
As we were using Cubic, we noticed that the following error kept popping up: "Error while dialing dial unix /run/zsysd.sock: connect: no such file or directory". However, the developer himself mentioned on the forum that the error can be ignored.

the kernel that will be installed. This is the kernel that the installer boots to rather than the kernel that is eventually installed to your hard disk. Occasionally, the stock kernel on a standard ISO won't work because of incompatibilities with your hardware, but an older or newer kernel will work. Install the kernel you want to work with on the Terminal page (`apt install <name of kernel>`) and then select it here if the latest one causes any problems.

'Preseed' allows you to alter software choices during installation. See the Debian Wiki (**https://wiki.debian.org/DebianInstaller/Preseed**) for a full guide to what preseeding is capable of and how it works. 'Boot' has options that mostly relate to the *GRUB* bootloader. In this example, we won't alter anything in the sections on this page. The next page allows you to alter the compression scheme used by *Squashfs*, and we'll leave this as is too.

Generate the ISO

The next page is the Generate page, and as soon as we go to this page, *Cubic* will start the work of building the installable system. In the interests of testing the system with a simple example, we'd recommend allowing this process to complete if you're new to using *Cubic*. Compressing the filesystem and building the ISO is quite a lengthy process, and usually takes several minutes or more, while giving your CPU and memory quite a workout. Linux always seems to use up a lot of memory when dealing with either large files, lots of files, or both; so consider shutting down unneeded applications at this point, unless you have a lot of system memory to spare. Things were a bit cramped on the 8GB machine we used for testing. If everything goes according to plan, the end result will be an ISO.



The chroot terminal of Cubic. You'll probably spend most of your time here, as this is where you can add extra packages and make other modifications.

Load the ISO into a virtual machine to complete the test. What you should be presented with is a standard Ubuntu installation, but it will install the extras that we have added (*GIMP* and *Inkscape* in this example). If you choose 'Test Ubuntu' rather than 'Install Ubuntu', you can use the live environment, and it too will contain the extra packages that we've added. If you select 'minimal installation' in the Ubuntu installer options, our changes will always be added, but the installation process will go faster and less cruft will be added to the installation.

Once the installation has completed, assuming everything has worked as it should, you should be able to boot into Ubuntu Linux and test it out.

Having gone through a test installation, we've only scratched the surface of what you can do with *Cubic*. Here are some further refinements and ideas to personalise the installation. To make the additions, you can open the *Cubic* working directory that you used before. This saves time and keeps the changes we made in terms of updating the package system.

Desktop environments

We can add the LXDE desktop environment and the LightDM login manager by typing `apt install lxde lightdm` on the terminal page. When you do this, you should see a text mode dialogue that gives you the option of choosing your default login manager. Choose LightDM. We now have to edit a text file to make LXDE the default desktop environment for new users. Within the *Cubic* chroot terminal type `nano /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf` . Apologies for the long filename there; it's not so bad if you use Tab completion to get to it. In this file, change the part after the `user-session=` to read `LXDE` rather than whatever is already in there. This means that you will now automatically log into LXDE, and in addition, LXDE will be the desktop environment of the live ISO, if you choose 'Try Ubuntu' rather than 'Install Ubuntu'.

It's possible to mass-copy installed packages from an already installed system using an old *apt* trick. Type `dpkg --get-selections > package_list.txt` on a running setup to gather a list of all packages installed on the system. You can then apply this list within the *chroot* terminal of *Cubic* by typing:

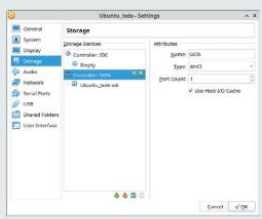
```
dpkg --set-selections < packagelist.txt
```

» LIVING IN A VIRTUAL MACHINE

One downside of creating a respin is that you usually need to carry out the installation, the longest part of the respin process, over and over again. Typically, you'll do this with a VM and any increase in efficiency here is well worth it.

As with all file-based Linux work, increasing the amount of available memory will speed things up. Allocating 2GB is about the minimum for reasonable performance. Allocate as many CPU cores as you can as the Ubuntu installer will make the most of them. If using *VirtualBox* as your virtualiser, there's an option in Settings... > Storage > Controller: SATA called 'Use Host I/O Cache', and we've found that it greatly speeds up tasks such as installing a distro. There's a slight risk of data loss within the VM if it crashes when using this option, but it's not a big deal with a job like this as you could just start the installation again.

Even if you plan to use the full install eventually, select 'minimal' in the Ubuntu installer to speed things up when testing.



VirtualBox (and possibly other VM managers) can speed up host I/O.

The package removal page. If ticked, these packages will not be added to the final Linux installation.

```
apt-get dselect-upgrade
```

You can also use this technique to 'save' the package list of a system that you've customised in *Cubic* by typing that first command into the chroot terminal of the installation. Of course, you can prune this list by hand in a text editor, but be warned, it's long!

Users and wallpapers

Whenever a distribution like Ubuntu creates a new user it creates a directory for that user and populates it with the contents of the **/etc/skel** directory. Let's look at a case where you want to give the user a custom backdrop image as soon as they log in for the first time.

The snag is that different desktop environments use different methods to display the background image. This is a method that will work for LXDE. In the case of LXDE, the file manager (*PCManFM*) draws the backdrop. To begin the process try customising the LXDE desktop in a running VM, and set the backdrop image to something in the **/usr/share/backgrounds/** directory. Having done this, copy the configuration file from its location within the home directory (**~/.config/pcmanfm/LXDE/desktop-items-0.conf**). The parameter within this file happens to be `'wallpaper='` and you can edit it by hand if you want to.

On the other side, copy the files to the filesystem of the *Cubic chroot* environment using the 'copy' icon at the top of the Terminal page. Place the image in the same place as before by typing `cd /usr/share/backgrounds/` and using the copy icon at the top.

Recreate the **config** directory structure and move into that directory with:

```
cd /etc/skel
mkdir -p .config/pcmanfm/LXDE
cd .config/pcmanfm/LXDE
```

Following this, copy the **desktop-items-0.conf** file into this directory using the copy icon.

There's quite a lot of potential here to pre-customise the user environment using the method of customising in a VM and then copying the configuration files. For example, let's say that you were producing a custom installation ISO for a college. In such a case, you might place a welcome pack of useful files (PDFs, images, etc) into the home directory. To do this, just place those files into **/etc/skel** using the *Cubic* interface.

All of the splash screens, such as the startup screen, used by Ubuntu Linux use a system called Plymouth. Customising Plymouth is a lengthy topic in itself as there are so many files that you can modify, and together these constitute a theme. The easiest way to get started with customising the splash screens is to browse the Plymouth themes at **www.gnome-look.org**. Most come with an installation script.

To use with *Cubic*, download the theme and unpack it and then copy the files to the *chroot* environment using the 'copy' icon. At the *Cubic* terminal, `cd` into the directory and run the installation script. Once this has completed, do `rm -rf [name of directory]` to remove the directory with installation files so that it isn't copied to the installation ISO. See the official Ubuntu Wiki (**https://wiki.ubuntu.com/Plymouth**) for more information on customising your own Plymouth themes. Most of those instructions don't require any modification to work under the *Cubic* chroot, but having made the changes type `update-initramfs -k all`.

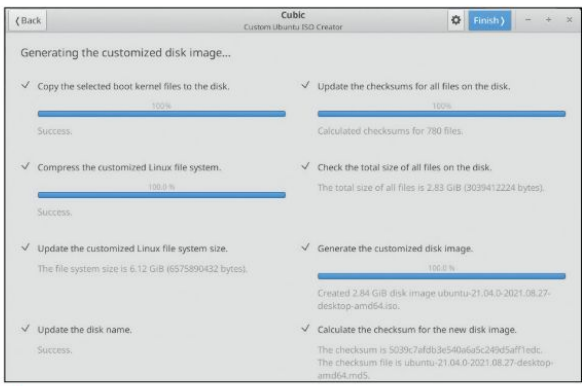
Custom PPAs

You can add debs and PPAs to Ubuntu in the normal way that you'd expect, using the *chroot* environment on the Terminal page. So, for example, you could add the *Inkscape* PPA and install *Inkscape* by typing:

```
add-apt-repository ppa:inkscape.dev/stable
apt install inkscape
```

This means that *Inkscape* will now be installed from the get go, and when you update the system, the updated versions will be pulled from the PPA rather than the Ubuntu repository. Do something like `apt-cache policy` and cut and paste that into a text file if you want to keep a complete list of every repository you've added to the system as a reminder for future respins.

Add downloaded debs to the *chroot* environment and install them with `dpkg -i [name of .deb]`. Nearly everything will work, but occasionally something requires a service that the *chroot* environment can't provide. As is the case with customising the user home directory, as detailed earlier, if you can't automate the installation, you could copy the .deb files manually and add a small post-install script, to be invoked manually, to install them. Happy respins! LXF



The ISO generation screen. This is a time-consuming process and memory and CPU usage will peak while it's going on.

GIMP

Credit: https://www.gimp.org

Improve your shots using optical filters

Mike Bedford looks at the benefits of photographic filters and how to emulate them – but some effects can only be achieved optically.

OUR EXPERT

Mike Bedford
Mike does have a few real optical filters which he uses to effect, but prefers the digital alternative whenever that's a sensible option.

Here we're going to be looking at photographic filters – but first, a word of explanation. The term 'filter' has taken on a different meaning in recent years. If our quick Google search is representative, a filter provides a means of digitally altering a photo. Commonly requiring little more than a single click, such filters can be used for artistic effect or for pure entertainment.

Before the advent of digital photography, though, a filter was a device that screwed onto the front of the camera's lens, and applied some effect optically. These effects were usually not nearly as extreme as those offered by some digital filters, but enabled film-based photographers to improve their work substantially. What's more, the benefits of optical filters or their digital equivalents can be just as important today.

In this tutorial we're considering only those effects that can be achieved optically, and so we've excluded the more bizarre effects that really don't need much in the way of instruction. Paradoxically, using filters to achieve more subtle effects often requires more understanding, but those more refined effects often make the difference between a photo that's just okay and one that's really good. What's more, in many cases, a really good photo is often one that doesn't appear to have been edited or filtered in any way.

Optical or digital?

First of all, we need to address the question of which is best: an optical or digital filter. As a general rule, we advise against applying any effect at the time you take the photo, and this applies not only to screw-on optical filters but also to in-camera or in-phone effects such as black and white or sepia conversion. The reasoning is simple. If your original photo has some effect applied, it will usually be difficult or impossible to remove that effect later by photo editing if you decide you don't like the result. If you avoid applying any effects while you're shooting, though, you have the option of applying whatever effects you like afterwards, while always keeping the original intact.

Despite this general recommendation, we need to point out that some effects that can be achieved with an optical filter can't be duplicated digitally. In this case, you'll need to use a physical screw-on filter and we'll cover these types of effect also. After all, despite our commitment to all things digital, it would be a questionable approach in our coverage of filters to avoid discussing several useful techniques just because they're optically-based.

All filters were once circles of coloured glass and, although some of these still have a place, most can be emulated digitally.

CREDIT: Kallemax, https://commons.wikimedia.org/wiki/File:55mm_optical_filters.jpg

Contrast enhancement

Some of the first and most widely used filters formed an important part of the black and white landscape photographer's arsenal. They were coloured yellow, orange and red and had the effect of darkening blue skies to increasing effects, thereby providing more contrast with white clouds. Similarly, green filters were used in portrait photography for darkening skin tones. These effects can be duplicated digitally, as we discovered in our tutorial on black and white photography in **LXF274**, so these optical filters are now really only for the traditionalist.

Filters have also been used in colour photography to enhance skies, but with a difference. The sky is usually much brighter than the ground, and this can cause problems. In particular, if the rest of the shot is correctly exposed, the sky will often look somewhat washed-out and, in the most extreme of cases, it will be overexposed so clouds could almost disappear.

One way of addressing this digitally is to take two photos, one with the ground correctly exposed and one with the sky correctly exposed. These can then be combined to provide the best of both worlds – see **LXF257** for information on HDR techniques. However, because this requires two shots of the same scene to be taken, it's necessary to use a tripod, and moving objects can cause problems, so this isn't for everyone.

This is where the graduated neutral density filter comes in. This is a filter that is clear at the bottom and grey at the top, the two regions merging one into the

QUICK TIP

One of the most extreme optical filters removes all visible light, allowing only infrared to pass through to the camera's sensor. This provides the option of infrared photography which has been described as otherworldly or even spooky. If you want to know more, see our tutorial in LXF248.

Using or emulating a graduated neutral density can revive a washed-out sky, and even make it look very dramatic.

other, and which can be moved up and down and/or rotated so that the division between the clear and grey regions line up with the horizon. It might seem that this can be duplicated digitally, and in some cases it can. However, this doesn't apply where the difference in brightness between the ground and the sky would cause the sky to be totally overexposed, because there's no way that can be reversed. In the most extreme cases, therefore, this is really something that should be accomplished with an optical filter. If you didn't use a filter when you took the photo, though, by all means see what you can do by photo editing – here we see how to do that in *GIMP*.

Having opened your image, make a copy of it as another layer by selecting Layer > Duplicate Layer. The duplicate is the topmost one. Now choose Add Layer Mask... from Layer > Mask and then, in the dialogue box, select 'Black (full transparency)' before clicking Add. Select the original (bottom) layer and choose Colors > Brightness-Contrast... and in the dialogue box reduce the brightness slider until the sky is to your liking. You'll probably want to increase the contrast too – which you can't do with a graduated optical filter. Don't worry if the ground gets too dark.

Now select the transparency mask of the duplicated top layer and paint over the ground in white so it becomes opaque and the non-darkened ground from the top layer shows. Zoom in and/or use a non-hard paintbrush when you're painting close to the horizon. An advantage over the optical alternative is that the delineation between the ground and the sky doesn't have to be straight line.

While on the subject of darkening skies in colour photos, further improving the contrast between sky and clouds, we really must talk about polarising filters. Because light from the sky is polarised, but light from clouds and most other objects isn't, by rotating a polarising filter you'll find a position at which the sky is notably darkened but most of the rest of the shot is unchanged. This can be achieved digitally, in much the same way that we saw for graduated neutral density filters. This time, though, instead of adjusting the brightness and contrast of the sky, go to Colors > Curves and adjust the line to the familiar S-shaped curve with the Blue channel selected.

Artistic effects

The filters we've encountered so far produce effects that could be described as enhancements of various types, but optical filters were also used for artistic purposes – and subtle artistic optical effects can be put to good use. However, our advice is not to overdo it; use these techniques sparingly.

The first is like the graduated neutral density filter but, instead of being partially transparent and partially grey, this is a graduated filter in which they grey portion is replaced with a coloured region. Commonly these are used, just like the graduated neutral density filter, for modifying the sky while leaving the ground unaffected. So, for example, a graduated blue filter can enhance a blue sky (or the sea if you use it the other way up), or a pink or orange one can mimic a sunrise or sunset. Loads of other colours are available, though, and the applications are limited only by your imagination. Needless to say, though, these effects can largely be duplicated in post-processing.

Mild soft-focus filters were popular for female portraiture, and this effect is trivially simple to duplicate

QUICK TIP

Some optical filters produced quite extreme vivid that might seem to be the sole domain of digital manipulation. Included here were multiple image and kaleidoscopic spiral filters; we even came across a rainbow filter. Many of these are still available, although duplicating the effect digitally would be preferable.

» BUYING FILTERS

Optical filters are defined by the size of their thread, and this must match the thread on your camera's lens. Excluding expensive professional lenses, this might vary from perhaps 27mm with compact cameras to upwards of 58mm for DSLR lenses. However, if you have more than one camera, or multiple lenses for your DSLR, you don't have to buy multiple sizes of each filter. Instead you can just buy filters to match your largest lens and add some stepper rings which enable you to fit filters to your smaller lenses.

This applies to the traditional circular filters, but rectangular filter systems might also be of interest. The basic kit comprises a holder to which you fit a rectangular filter, and one of several sizes of screw adapters. This is another way of using a single filter with different lenses.

Most phone cameras don't have a filter thread and this is true of some compact cameras. However, this doesn't necessarily mean that you can't use a filter. The solution in this case is to use an adapter. For phones, this often takes the form of a filter thread with a clip that attaches to the phone; for compact cameras a common form of adapter attaches to the camera's tripod socket. Although not recommended for regular use, at a pinch you could even hold the filter just in front of the lens.

QUICK TIP

Apparently identical filters can differ drastically in price, and you get what you pay for. The law of diminishing returns applies but, if you have a good camera, it would be a good idea not to go for a bargain-basement filter. Also, do make sure your filters are clean, because a dirty filter could easily degrade your photos.

digitally. A related effect is the soft-focus spot filter that keeps the centre of the image sharp but causes the edges of the photo to be blurred, sometimes substantially so. This is also easy to do in *GIMP*. To start, you need to duplicate the first few steps of our earlier *GIMP* example, duplicating the image as a new layer and applying a fully transparent mask to that layer. Next, with the bottom layer selected, choose Filters > Blur > Gaussian Blur... and in the dialogue box adjust the Size X and Size Y sliders until the photo is adequately blurred. Of course, the whole photo will blur.

Finally, with the transparency mask of the top layer selected, paint a large circular or oval white object so you see a sharp area in the centre. To get it right you'll need to adjust the size, aspect ratio and hardness of the paintbrush and fine-tune the position, undoing the action and trying again if it doesn't look quite right.

Next we're going to look at adding a starburst to highlights in an image, an effect that can be quite effective and was traditionally achieved with an optical filter. However, it can also be duplicated digitally. Remember that this effect will most definitely lose its appeal if you use it on all your nighttime shots. We'll leave you to try out *GIMP*'s options, but basically you need to go to Filters > Light & Shadow > Sparkle...

This is a case where less can often end up providing more, so don't wind up the settings too far. It's quite possible that you won't be able to fine-tune this effect perfectly, so you might end up with more starbursts than you really want. However, this is easy to fix using a similar technique to the one that we've already seen of duplicating the image as a second layer with an added transparency mask, applying the effect to the bottom layer, and adjusting the transparency of the top layer as necessary.

GIMP provides plenty of options for applying a starburst effect to highlights, but you can really overdo it if you're not careful.

CREDIT: CC0 1.0, https://www.maxpixel.net/Railway-To-Train-Train-Railway-Line-Transport-3087855.

Optical-only effects

You might find it surprising, but some optical filters do useful jobs that can't be duplicated digitally. First is the polarising filter; while its effect of darkening skies can be achieved in *GIMP*, there's another effect of a polarising filter than you can't achieve by post-processing. Because light reflecting off the surface of still water or a window pane is polarised, you can use a polarising filter to eliminate, or at least reduce, the reflected light. The reduction isn't always 100 per cent but it can make a huge improvement as our 'before and after' photos demonstrate (see opposite page). In the case of reflection off the water in a lake, it really doesn't matter too much if a small degree of reflection remains. Most such filters are circular polarisers (rather than linear) and this is a must with modern cameras, because otherwise autofocus might not work properly.

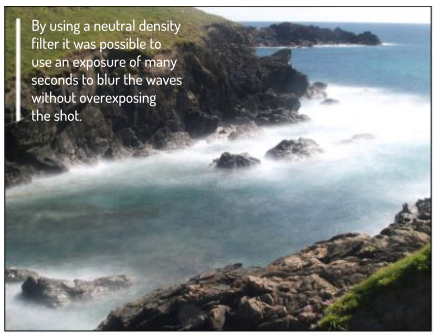
Another type of optical filter that you can't emulate in *GIMP* is the neutral density filter. This is different from the graduated neutral density filter that we've already seen, in that it reduces the light reaching all parts of the photo, not just the sky. At first sight this might seem an odd thing to do, since a common photographic principle is to maximise the amount of light. While plenty of light gives the option of a good depth of field while minimising the shutter speed and hence avoiding camera shake, there are occasions where you want a

Emulating a soft-focus spot filter can produce an appealing effect, and doing it digitally offers so much more flexibility than with a screw-on filter.

» FILTER FACTORS

If you're going to be using real optical filters, you ought to be aware of the concept of the filter factor. Your camera's metering system will take it into account, but it pays to bear it in mind.

Most filters reduce the amount of light entering the camera. With a neutral density filter that's the whole reason for using it, but with other filters it's a side-effect that will affect the exposure. This reduction is expressed as a so-called filter factor. A filter factor of 1 means that there is no reduction in light, 2 means that the light is reduced by a factor of two (ie. 1 stop), and so on. To give some examples, a skylight filter has a filter factor of 1.1 while with a polarising filter this increases to 3 or 4. The light-reducing effect of a polarising filter is quite substantial, therefore, and cannot be ignored. Since that four-fold reduction in light would have to be compensated for either by increasing the aperture by two stops or increasing the exposure time by a factor of four, you'll have to take it into account. For example, if you want to maintain the depth of field so that increasing the aperture isn't an option, this might tip the balance towards using a tripod.



By using a neutral density filter it was possible to use an exposure of many seconds to blur the waves without overexposing the shot.

long exposure. A classic example is in scenes with moving water such as a waterfall, so the water is blurred. On a bright day, though, you might not be able to use an adequately long exposure without overexposure. If you want an exposure of a few seconds – as needed, for example, to make waves on the sea appear almost like a mist – the problem gets worse.

Neutral density filters provide an obvious solution here. They're classified according to their strength, ND2, ND4 and ND8 being common, reducing the amount of light by a factor of 2, 4 or 8 respectively. You'll probably want to buy several filters of different strengths, and they can also be used in combination, so with ND2, ND4 and ND8 filters you have the option of reducing the light by a factor of 2, 4, 8, 16, 32 or 64. Another option is to buy a variable neutral density filter.

Probably one of the most common types of optical filter is the so-called skylight filter, otherwise known as a 1A filter. At first sight it might look like a circle of ordinary glass and its effect is certainly subtle. In fact it's a very faint pinkish colour and it filters out ultraviolet and a very small amount of blue light. The latter was to counter a blue cast on some colour films, while the former can reduce haze, especially at altitude.

While cutting the blue content is no longer required – and is barely noticeable – cutting haze is still valuable, and can't be achieved digitally. However, there's another reason for the popularity of the skylight filter, namely to provide protection for the lens. While these filters are not as popular as they were in the era of film, the value of protecting your lens is obvious. If you want to achieve that with no visible effect at all, so-called protector filters are available.

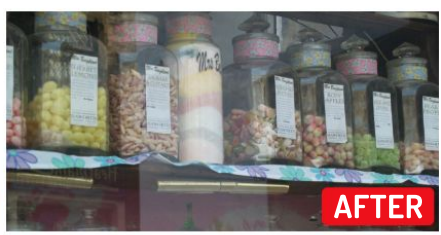
Strictly speaking, the final filter we'll mention isn't a filter at all, but a lens. It's often thought of as a filter because it screws onto the camera's filter thread. The close-up filter does just want the name suggests, allowing you to get closer to your subject than your camera's minimum focusing distance. They're available in various strengths which increasingly reduce the minimum focusing distance while, at the same time, reducing the field of view but increasing the magnification. Common varieties are +1, +2, +4 and +8 and you'd be advised to read up on how each will modify your camera's minimum focusing distance and field of view before deciding which to buy. They're commonly available in sets and, since they can be combined, buying a set will offer you plenty of flexibility. LXF

» SUBTLY COLOURED FILTERS

We've touched on the dark-coloured yellow, orange and red filters that were used in black and white photography, but a huge range of other coloured filters are available, for example the Kodak Wratten range. Paler blue and orange filters were used in the days of film to allow daylight film to be used with artificial (tungsten) lighting and vice versa, but they aren't needed in digital photography thanks to auto white balance. However, there are also even paler filters that might be of interest.

Those filters were used to make a shot slightly warmer or slightly cooler. Digital cameras will cancel out any such colour shift with their auto white balance – unless you shoot in RAW format – but the same effect can, of course, be achieved in post-processing. We read less about this element of photo-editing than many other topics, but the technique can be quite effective so it might be worth experimenting with it. In *GIMP*, while there are several ways of achieving this, the most intuitive is via the Color Temperature dialogue that you'll find in the Color menu. Simply increase Original temperature to make the photo cooler, or decrease it to make it warmer.

Warming or cooling a photo once required you to invest in a selection of coloured filters. With GIMP, all it takes is a slider.



If you fit a polarising filter to the lens and rotate it until the optimal position is found, reflections can be very much reduced.

LINUX FOR THE SOUL

Jonni Bidwell wants a free, enterprise-grade operating system and word on the street is AlmaLinux is the coolest one going…

Our "Faster Better Servers" feature in LXF278 attracted more than the usual number of letters, and they weren't solely about the glorious racing car imagery. Some called for a more practical and fairer treatment. OK!

Red Hat Enterprise Linux takes security seriously and the ecosystem is rich with tried-and-tested tooling. This is all inherited by the communities downstream (such as CentOS, Rocky Linux and AlmaLinux) making any one of them a fine choice for your Faster, Better servers. It would be good to explore that. Thousands of machines at CERN and Facebook ran CentOS, and perhaps you did too. CentOS 8 will see its support period expire at the end of 2021, as Red Hat concentrate their efforts on CentOS Stream. So some might still be looking to migrate.

And those that have migrated away from CentOS may be questioning their choices. Fortunately, a product of CentOS 8 and any of the new projects to emerge as a result of its demise being 1-1 binary compatible with RHEL is that they are 1-1 binary compatible with each other. So it should always be straightforward to switch between them. CentOS Stream, Red Hat's community offering that will be the proving ground for future RHEL releases, isn't going to be every CentOS Original user's cup of tea. But who knows which of the new distros will stand the test of time, or what direction they'll go in.

Another point of comment was that we really only covered CentOS Stream and Rocky Linux, set up by CentOS cofounder Greg Kurtzer and named after fellow co-founder Rocky McGough. These are by no means the only places for CentOS users to go. And we were perhaps remiss in not giving more attention to alternative RHEL downstreams. In particular AlmaLinux has established itself as a serious contender.

Alma takes its name from 'soul' in many languages, a nod to the 'soul' of Linux being its community. As such, it aims to be governed and guided by the community, very much in the spirit of the original Community ENTerprise Operating System. It's also an enterprise grade OS that deserves to run on your servers, so we'll show you how to set up an AlmaLinux virtual machine in a jiffy with Vagrant.

Back in LXF182, in a corner of Dr Brown's Administeria cupboard, was some prescient comment from the good doctor. It was 2014, and Red Hat had just announced its partnership with CentOS. Red Hat CTO Chris Wright was sceptical of Red Hat's motivations. It promised great build infrastructure so the community could develop their own CentOS downstreams, begging him to question, "Wasn't Fedora meant to be where new technology slated for inclusion in [RHEL] was tested. Doesn't that work anymore?"

It's not immediately clear how the partnership helped CentOS – the CentOS-RHEL development firewall would remain, so CentOS would still be a month or so behind its upstream. And four of its key developers were now Red Hat employees. And their logo was now an official Red Hat trademark.

Seven years later Mike McGrath, Red Hat's own VP of Linux engineering (in an interview with *IT Pro Today*), said that, "CentOS itself was not actually providing that much usefulness to Red Hat. Most of the communities we set up – Fedora, for example – do have a lot of bidirectional community involvement. Unfortunately, CentOS was never like that. It was always a community of users, so that contribution model was mostly one-way."

Filling the gap

And it's easy to see how its new offering, CentOS Stream, aims to do better. As Red Hat senior community architect Karsten Wade explains: "Essentially, Red Hat is filling the development and contribution gap that exists between Fedora and RHEL by shifting the place of CentOS from just downstream of RHEL to just upstream of RHEL."

Chris Wright said when CentOS Stream was introduced that "developers need something more to address their specific challenges. They require earlier access to code, improved and more transparent collaboration with the broader partner community, and the ability to influence the direction of new RHEL versions". At that time (September 2019) no one had mentioned anything about CentOS being curtailed, but surely some had their suspicions.

The AlmaLinux team spent a lot of time wondering out how it had all gone wrong for CentOS. Back in September, Community manager Jack Aboutboul summed it up in an interview with *The New Stack*. "I've spoken with CentOS maintainers in the past and I remember just how overworked and understaffed they [were]," he said. "Ultimately, people would just get tired; it's a lot of tedious work with no replacement, and no end in sight." A little like magazine production then (we jest, of course).

Alma Mirabilis! Seeking to stir the souls of CentOS refugees.

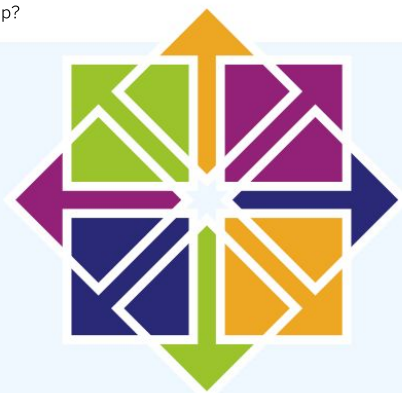
> **JACK ABOUTBOUL ON CENTOS**
> "I've spoken with CentOS maintainers and I remember how overworked and understaffed they [were]. It's a lot of tedious work with no end in sight."

Besides developer fatigue, the other obvious problem was how Red Hat was able to acquire all the CentOS trademarks in the first place. This was largely a result of that project's own governance. Finally, there was the question of longevity – even if someone makes the fastest, bestest (*sorry Cliff–Ed*) free RHEL clone, who will maintain it in the future. What happens when sponsorship dries up?

WHAT'S IN A REBUILD?

Building CentOS was hard work. This is an important point. It's easy to say, "Well, you just grab all the source RPMs, remove the branding and build them." But that's not so simple.

In the early days – before there was a formal build infrastructure – packages tended to be built on maintainers' home PCs. This was tedious work, and it also made onboarding new CentOS developers tricky. And the build process for CentOS itself was, at least for a time, not so well documented. CentOS always strived to not patch anything that would take it further from RHEL than it needed to be. But sometimes as a result of errors in Red Hat's own packaging, such extra patches were a necessity. And, for a time at least, talk of rebuilding CentOS was considered taboo on its own mailing list. "It's built for a community... not by a community," said Johnny Hughes back in 2011. All of which makes the development process look rather closed in hindsight. So even before Red Hat became officially involved, things were not all rosy.

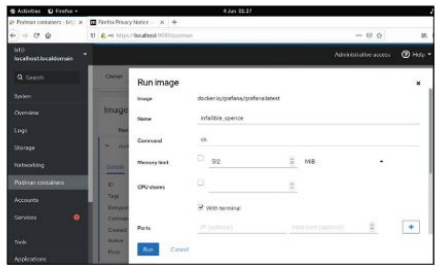


In retrospect, the pre-Stream CentOS community featured a lot of forces pulling in different directions.

It's important to note there have always been other RHEL clones besides CentOS. Tao Linux folded into CentOS in 2006. Likewise, White Box Enterprise Linux. Startcom Linux didn't fare much better, either. Scientific Linux, developed at CERN, now faces an uncertain future, though version 7 will be supported as planned until 2024. Oh, and Amazon and Oracle Linux are both derived from RHEL, and they probably won't be going anywhere soon. There's also CloudLinux OS, a premium operating system designed for hosting. The CloudLinux team has been rebuilding RHEL since 2010, and it's CloudLinux that are behind the AlmaLinux effort. As Jack says (in the same *New Stack* interview), "We recognised that we were in a unique position to give back to the community on the technical front and to make sure the mistakes of the past with the ownership wouldn't and couldn't happen again".

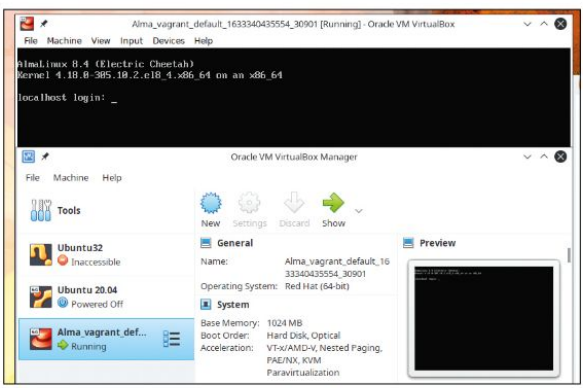
Those issues of governance are addressed by making the AlmaLinux Foundation a 501c6 non-profit with an independent board of directors and communal ownership. Contributors have voting rights, so in essence everyone has a voice. This seems much more in the spirit of community than the "community of users" that CentOS unwittingly fostered.

So AlmaLinux gets a big tick for wholesome ideologies. But where does it stand technically? Well, it's already received sponsorship from Amazon and Microsoft, so you can try it out yourself in the AWS (**https://bit.ly/lxf282almaaws**) or Azure (**https://bit.ly/lxf282almaazure**)clouds. Or you can go small and try out its Raspberry Pi (**https://github.com/AlmaLinux/raspberry-pi**) image, bringing enterprise stability to the Pi-verse. To finish off this feature though, we'll look at something in between: running AlmaLinux as a virtual machine, but set up using the ever-fashionable *Vagrant* provisioning software from HashiCorp.



Administration with Cockpit is easy, and container orchestration with Podman is simple.

Vagrant makes it possible to set up custom virtual environments on any platform. It automates the otherwise tedious process of manually setting up a VM (or several if you like), and since everything is specified in a text file (a Ruby script called `Vagrantfile`), the resulting machine(s) is highly portable.



From a single command line incantation this whole VM sprung into life.

Get started with Vagrant

Our first step will be to install *Vagrant*. On Debian (or Ubuntu) this is a matter of:

```
$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
$ sudo apt-get update && sudo apt-get install vagrant
```

You'll find instructions for other operating systems on the *Vagrant* website at **www.vagrantup.com/downloads**. *Vagrant* works with a variety of virtualisation platforms including *VirtualBox*, *KVM* and *VMware*. *VirtualBox* is the default and probably also the most familiar to many *Linux Format* readers. So let's go with that one (we'll assume you've got it installed already).

Prebuilt *Vagrant* boxes (so-called base boxes) for

UNIVERSAL BASE IMAGES

In 2019, to shake up the container world. Red Hat introduced Universal Base Images (UBI). These aim to solve the problem of containers not being portable, since they have to be rebuilt whenever they're deployed elsewhere. Red Hat's UBI images aimed to define standard package sets and runtime languages to make container builds repeatable across deployment targets.

Now, thanks to AlmaLinux's Cloud and Containers SIG (Special Interest Group), UBI-compatible images are available for AlmaLinux, too. They offer four variants: Base, Micro, Init and Minimal. The latter uses *microdnf* instead of the full fat *DNF* package manager, so doesn't need Python to run. When it's built it occupies only about 100MB. Micro is even smaller, and doesn't even have a package manager. Fewer moving parts means less room for vulnerabilities to creep in, so this image is touted as offering increased security. If you need more flexibility, the init image can be used to containerise multiple applications at once and manage them all with the init system of your choosing.

Container images are available for x86_64 and arm64 architectures, and you can see how they're made at **https://github.com/AlmaLinux/docker-images**. For more information on the Cloud and Containers SIG, check the wiki at **https://wiki.almalinux.org/sigs/Cloud.html**. That wiki is always expanding, so check out the other pages, too – there's probably something a talented individual like you can help with.

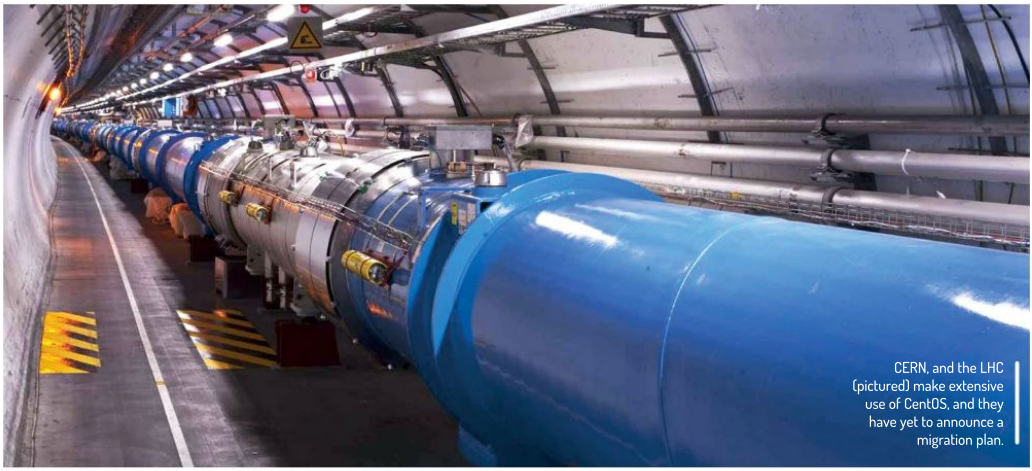


Image credit: CERN CC-BY-SA)

CERN, and the LHC (pictured) make extensive use of CentOS, and they have yet to announce a migration plan.

AlmaLinux are available for download on the official *Vagrant* Cloud website and also at the community site (**www.vagrantbox.es**). To understand how these work, it's good to first build from the ground up. To add the official AlmaLinux box to *Vagrant* follow these steps:

```
$ mkdir alma_vagrant
$ cd alma_vagrant
$ vagrant init almalinux/8
```

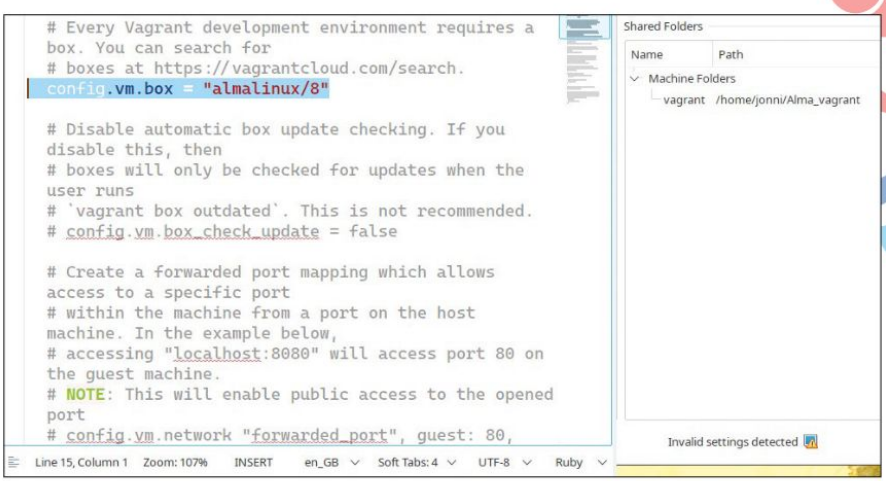
The final line creates a **Vagrantfile** in the current directory, so it's good practice to keep your configurations tidy. If you look at this file, you'll see most of it is commented out. All but the `config.vm.box` directive in fact, which refers to the box id we specified on the command line. Looking through the comments though, you'll see it's easy to set up things that you'd have to do manually in *VirtualBox*. Private networking, port forwarding and synced folders can easily be predetermined. You can even add inline shell scripts to, for example, update the system and install Apache.

We'll leave finicky configuration details for now and go right ahead and fire up our virtual Alma box:

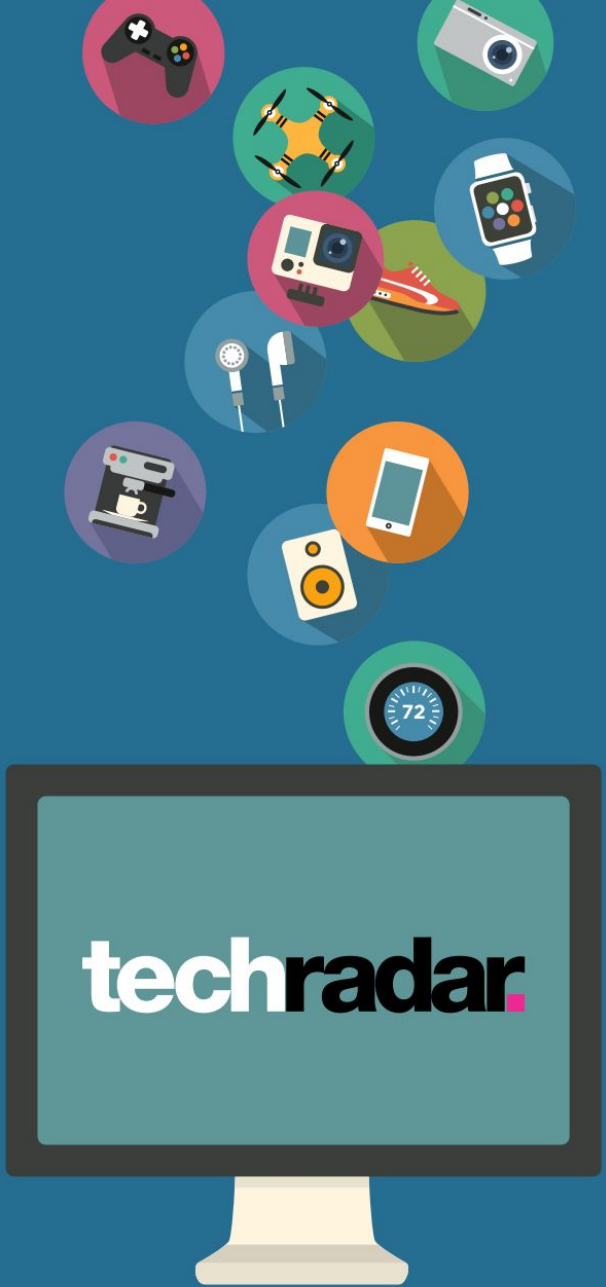
```
$ vagrant up
```

This will download the actual machine image (which is around 700MB) and rather verbosely tell you about everything it's doing. In particular you'll see that port 22 on the guest is forwarded to port 2222 on the host, enabling you to SSH into the VM. You'll also see that the **/alma_vagrant** directory we set up earlier, is mounted inside the guest at **/vagrant**. If you go ahead and fire up the *VirtualBox* GUI, you'll see that the machine has indeed booted and is running. But close that clunky GUI – you can do all the required administration from the command line.

To SSH into the machine just run `vagrant ssh` from the same directory. We can't readily do this manually because the SSH key set up during provisioning is buried deep within Vagrant's configuration. To continue with your own configurating, since we've unexpectedly run into the end of the page, look at the excellent documentation on the Vagrant website.



The highlighted line is the cornerstone of our Vagrantfile, but just about everything can be configured from here.



The home of technology

techradar.com

HotPicks

Czkawka » Cozy » Tomatoid5 » Scrcpy » VMPK » Biblioteq » Timefind » SuperTuxKart » Fheroes2 » Reptyr » Hover Zoom+

Alexander Tolstoy has a secret trick for correctly spelling the name of this month's opening *HotPick*...

DUPLICATE FILE REMOVER

Czkawka

Version: 3.2.0 **Web:** https://github.com/qarmin/czkawka

Sooner or later most of us will experience a hard drive running out of space. This may be due to the ever-growing assets under the home directory, the abundance of downloaded Linux ISO files, or simply because of too many Flatpaks and Snaps that are known to devour disk space.

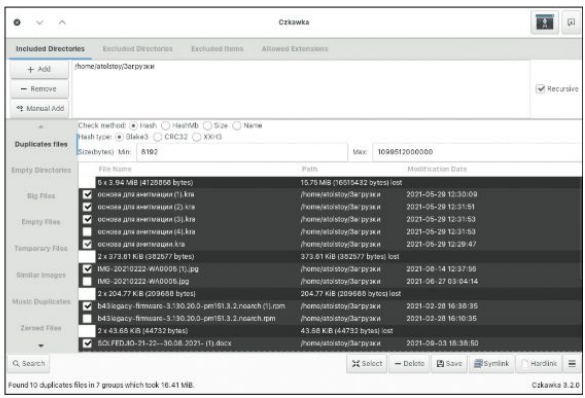
No matter the reason, it's time to figure out what exactly occupies so much space, and as such we're going to go a bit further then simply analysing disk usage with *Baobab* or *Filelight*. *Czkawka* is designed to track down unnecessary files. It enables you to remove file duplicates, large files, empty directories and other stuff that you probably don't need anymore.

The *Czkawka* window has lot of tabs and buttons, yet it doesn't take long to figure out how to use the program. Under the Included Directories column you can add at least one location where *Czkawka* will search for duplicates. Note that you can define how exactly the application will detect similar files: by name, size, or one of several hashing algorithms.

The left-most column in *Czkawka* has categories for other search targets including empty files and directories, big objects, temporary files, zero-size files, broken files and more. One of the more interesting options is Music Duplicates, which helps locate audio tracks that have identical tags. These are not necessarily file duplicates – you may have various versions and performances of the same songs in your music library – but it's still a useful feature.

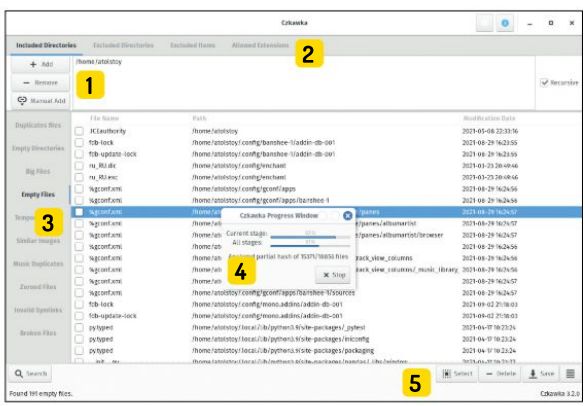
Whatever search target you decide to use, you can further fine-tune *Czkawka* and manually exclude certain directories or items, and even use the whitelist with permitted file extensions. Everything is completely safe since any search query is just a 'dry-run': *Czkawka* will only delete files after you explicitly select them in the search results panel and press Delete. If unsure, press Save and review the search results later on.

With such a solid set of features, *Czkawka* is hands down the best file sweeper available for Linux!



We bet you didn't know your /home dir had so many useless files!

GET TO KNOW THE CZKAWKA INTERFACE



1 Places to search
Add directories where the program will search for files. The Home directory is a good universal variant.

2 Fine-tune the scope
It's possible to exclude files or directories from searching, and also limit the scope by explicitly whitelisting allowed file types.

3 Not just duplicates...
Czkawka supports many search criteria and is able to detect oversized, zero-sized, broken, temporary and many other file types lurking within your system.

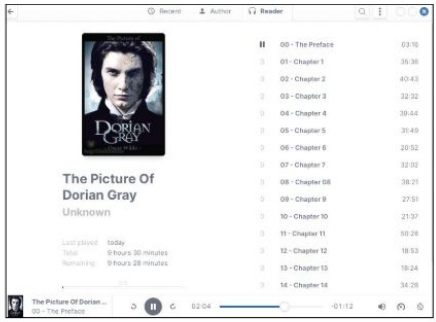
4 Perform the scan process
Large directories take a while for Czkawka to chew through, but at least you can keep an eye on the progress window.

5 Time to take action!
The left-most Search button triggers the scan process while the buttons to the right enable you to take action against search results: select, delete, save the list and more.

AUDIO BOOKS PLAYER

Cozy

Version: 1.1.2
Web: https://github.com/geigi/cozy



Populate your library with audio books and get the most of Cozy, the friendly and polite book reader.

We first treated *Cozy* as yet another audio player as many people listen to audio books with their favourite media player, such as *VLC*, and there's nothing wrong with that. However, *Cozy* ships with a couple of features that were designed specifically for books. You can rewind or skip forward parts of the book by a duration of your choosing.

Cozy is also equipped with a sleep timer. The idea is that if you listen to a book for a long time and eventually fall asleep, *Cozy* will automatically stop after the chapter is over. Additionally, it can suspend or power off your Linux computer when this occurs.

Cozy makes a great effort to gather end users' feedback. When you first launch the application it asks if you want to provide reports in case of errors, and how detailed such reports should be – drag the slider to set the appropriate level.

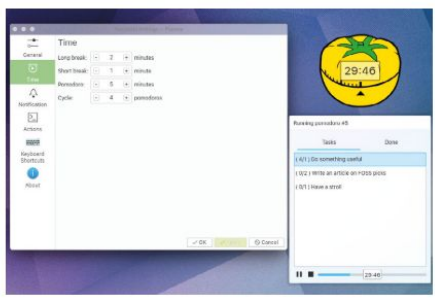
Next, *Cozy* takes you to the main screen and suggests that you may want to either drag an audio file over the program's window, or specify the directory where your audio books are stored. The application leaves a distinct feeling of being cozy – not just in its name but also in various details that show the commitment of the development team to improve the user experience.

The *Cozy* interface is optimised for storing lots of books, and much like an advanced music player it has dedicated features to run a music library. There are three main view modes: Recent, Author and Reader. Each mode lists thumbnails of your audio books, and we recommend to make sure you that you have the cover artwork available: either embedded inside audio files, or stored separately. Book covers enhance the whole Cozy experience, and they also help you identify a book more quickly. It's possible to store books in any folder, or in several different folders. *Cozy's* Preferences dialog makes it possible to add as many folders as you like.

PLASMA EXTENSION

Tomatoid5

Version: 0.1
Web: www.pling.com/p/1574941



Easily manage current and archived tasks, and adjust the work/rest periods with the assistance of Tomatoid5.

Have you ever found yourself unwilling to start an important project and instead have postponed it? If so, then you're suffering from a bad case of procrastination (*Oh, hi Jonni – Ed*). This is a mildly negative character trait that can be addressed by several techniques.

One such approach is the Pomodoro Technique, created by Francesco Cirillo in the 1980s. He suggested combating those self-initiated delaying tactics by breaking the time into periods of focused work, followed by breaks. A focused work period usually lasts for 25 minutes – something that anyone can measure with a kitchen timer, like the tomato-styled one Cirillo used himself. In this month's *HotPicks* we'll be implementing the Pomodoro technique using the *Tomatoid5* widget for the Plasma desktop without delay!

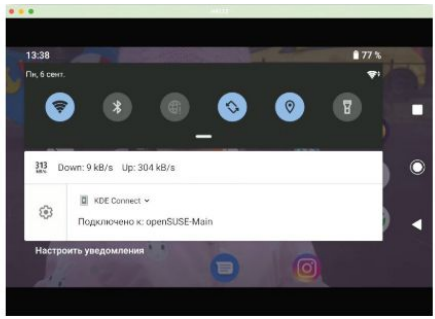
Tomatoid5 is the long-awaited Plasma 5 port of the original *Tomatoid* widget that used to exist for KDE4. It looks like a red tomato that you can place at any place on your desktop, inside a panel, or in the Latte Dock. Click the tomato to bring up the task list. Here you can add as many tasks as you wish, but for each one it's important to provide the estimated number of 'pomodoros' that you think will be required to complete the task. Each 'pomodoro' comprises 25 minutes of work and five minutes of rest. The fifth pomodoro assumes that you take a longer break of 20 minutes.

We found *Tomatoid5* to be a fully featured Pomodoro implementation for Linux. You can adjust the duration of periods and tell the widget if it should start a new 'pomodoro' after a break in auto mode. The presentation is also superb. When in the focused mode, *Tomatoid5* changes its colour to yellow and starts ticking. When it's time to have a rest, the widget turns green and displays the relevant desktop notification. Access the Settings>Time section to adjust the lengths of work/rest periods, and the maximum number of iterations that *Tomatoid5* will perform.

ANDROID CONNECTOR

Scrcpy

Version: 1.18 **Web:** https://github.com/Genymobile/scrcpy



View and control your Android phone right from the Linux desktop.

There are several technologies that help pair an Android smart phone with a Linux desktop. So far we can mount and browse the phones' contents when it's connected via USB, and enjoy the glorious KDE Connect features for remote control, messaging, shared notifications and more.

The only missing feature was screen sharing, but if you need to show the desktop screen on a mobile phone, you'll probably use *Deskreen* (**LXF274**). This time we'll be trying a program that does the opposite: it shares the Android screen for Linux. *Scrcpy* (short for 'screen copy') is a command-line utility with a solid collection of debug options and features aimed at Android app developers. However, it's also useful for mere mortals who just want to see their phone's screen without going away from their Linux box.

Scrcpy is included with most Linux distributions, so installation is straightforward. In order to run *Scrcpy* on the Linux side, make sure you have the *android-tools* (or similarly named) package installed. On the phone side the only requirement is the enabled remote ADB debug in the Android settings. No rooting, hacking or tweaking is required. Plug your phone with a USB cable, run `scrcpy` and tap OK on the phone to enable the connection. So far this is enough for instant screen sharing via USB. *Scrcpy* also works flawlessly via Wi-Fi, provided that you enable wireless debug. Before unplugging the USB cable, issue this command:

```
$ adb tcpip 5555
```

Then find out the phone's IP address:

```
$ adb shell ip route | awk '{print $9}'
```

Finally, unplug the cable and tell ADB to connect via WiFi with:

```
$ adb connect 192.168.X.X:5555 # use your real IP
```

Now run `scrcpy` as usual to see your phone's screen. *Scrcpy* not just displays, but also controls your phone and has a range of cool tricks, too!

MIDI GENERATOR

VMPK

Version: 0.8.4 **Web:** https://github.com/pedrolcl/VMPK

Enjoy a responsive virtual piano with dozens of instruments, sustain, note editing and more.

We're always pleased to see an open source tool compete with well-known commercial products. A good example is *Virtual Midi Piano Keyboard*, or *VMPK* for short. This is a spectacular application for musicians and also for everyone else who enjoys piano music. *VMPK* doesn't produce any sounds itself, but does provide a user-friendly interface between you and the MIDI subsystem in Linux. Naturally, those people who own real piano synthesisers will be most interested in *VMPK*. This tool is superb for digital music recording once you connect the piano keyboard to your Linux machine. Even the standard MIDI bank (aka 'General MIDI') has a vast selection of instruments to choose from, and you can attach custom banks, too.

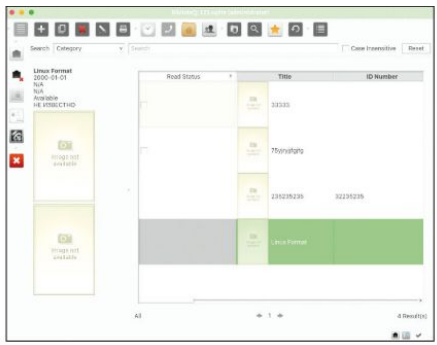
You can play on a digital synthesiser and have an editable MIDI recording, or edit an existing .mid file in *VMPK*. You even don't need to have a physical instrument to make use of *VMPK*. The standard piano keyboard responds to computer keys just fine – it's just a matter of setting key bindings to make playing piano using a QWERTY keyboard convenient. *VMPK* is configurable, and you can change key mappings under the Edit menu and then save or import existing configurations. Moreover, *VMPK* is capable of preparing multi-track compositions, adding drums, bass lines and more thanks to the SoundFont support. You can try out your musical skills right after *VMPK* launches.

There are many cases where *VMPK* can prove to be helpful as a MIDI generator and receiver. Apart from recording and editing music, it can help in online teaching by displaying the tutor's keyboard on the screen. And if you run *VMPK* on a device with a touch screen, you can learn how to play a piano right away. Search for *VMPK* using your favourite package manager.

ARCHIVING TOOL

Biblioteq

Version: 2021.08.08 **Web:** https://github.com/textbrowser/biblioteq



Fancy running a cool GUI for SQLite focused on storing books and disks? Then don't overlook Biblioteq!

When we write about archiving we usually mean data compression software, but in the case of *Biblioteq* we're referring to library software. Imagine that you've built up a collection of DVDs, magazines (*what now?!–Ed*), books, video games and other physical items that you want to catalogue and store – much like in a real-world library. *Biblioteq* is a feature-rich graphical shell for those who want to meticulously keep their collections perfectly organised.

From a technical standpoint, *Biblioteq* is a front-end to the SQLite database engine. You can build and maintain sophisticated SQLite databases with cross references and dozens of tables, all without any specific knowledge in database administration.

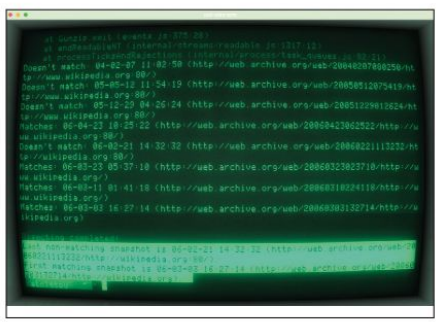
When you fire up *Biblioteq* it first asks you to connect to a database. You may not have any, so feel free to go to File>New SQLite DB and create a new database. Once you've done, connect to it and you're ready to go. Press the + button to create a new element based on suggested templates. Each one assumes that you need to fill in the item's card and provide a detailed description. Not just name, author/artist and year, but a dozen other pieces of information, including keywords, ISBN number, language, region, duration and more. And of course you can add covers, attach extra files and link other items to better track down a particular item in the future.

Biblioteq provides all the necessary options required to run a real-world library. As such, you can create lists of readers, countries, languages and other entities, manage links and cross-references, perform book reservations, search for items by their details and much more (including support for direct SQL queries to the DB). So, if you happen to run your own library, keep in mind that *Biblioteq* covers all possible requirements, and it's totally free, too.

WEB ARCHIVE TOOL

Timefind

Version: 1.0 **Web:** https://github.com/Cykelero/timefind



With Timefind we made sure Wikipedia had reached the one million articles count 15 years ago!

The internet remembers everything – so they say. In fact the web is more like a river and we all sit on a wooden platform that drifts with the current. We see what happens today and what had happened in the not-so-distant past, whereas older things often become swamped under torrents of content, or even lost forever. That's where the Internet Archive, aka the Wayback Machine comes into play. Of course you can go to its web site and manually roll back time to see how a web page used to look before, but we've got a tool that does it better. *Timefind* is a NodeJS-based utility that automates Web Archive searches in such a way that you can quickly find out the first and the last appearance of a word or a phrase.

Timefind flips through saved snapshots of the web site you choose and searches for valid occurrences of the string provided. To install the application make sure you have NodeJS set up and then fire up this command:

```
$ npm install -g timefind
```

Using *Timefind* is also easy. Here we try to find out the first mention of 'community' on the Elm page:

```
$ timefind elm-lang.org community
```

It's also possible to use regular expressions. Let's find out when Wikipedia reached one million articles:

```
$ timefind wikipedia.org -r '\\d \\d{3} \\d{3}'
```

Timefind's default behaviour is to look for a string being added, and then never removed, for the complete lifetime of the page. Therefore, *Timefind's* result is presented as a 'bisection' with two points: one for 'Last non-matching snapshot' and another for 'First matching snapshot'. Normally both refer to similar dates. The accuracy depends on several criteria, such as the site's popularity and the string occurrence on its main page. You'll probably get the best results searching inside well-indexed places of the internet, such as CNN or Wikipedia that have hundreds of thousands of snapshots available at the Internet Archive cache.

RACING GAME

SuperTuxKart

Version: 1.3 RC
Web: https://supertuxkart.net

Choose your new challenge by wheeling around the dedicated land of different stories and play modes.

SuperTuxKart is the best open source penguin-based carting game on Earth. We took a sneak peek at its release candidate and spent numerous hours playing this addictive "karter".

SuperTuxKart 1.3 RC has some new exciting additions. In particular, the new version features two new maps: Ancient Colosseum Labyrinth that resembles the Roman colosseum; and Alien Signal, which was inspired by the real-world location of the extra-terrestrial SETI programme. The new graphics settings under the Options menu make it easier to adjust the game to suit your hardware. *SuperTuxKart* can look drastically different depending on your video card. The low-graphics mode disables most of high-res eye candy such as dynamic lighting, particles and texture smoothing in order to maintain a comfortable FPS. If your graphics card is up to the job, feel free to max out the game's graphics settings.

First-time *SuperTuxKart* gamers may want to complete the in-game tutorial before they're dropped into the story-mode race. It's always good to first learn how to use power-ups, such as nitro bottles that enable extra acceleration for a brief time, look behind, or return to the track after accidentally getting stuck.

SuperTuxKart is all about cheating: collect the present boxes along the way and shoot your competitors, place obstacles on their way, or protect yourself from their attacks using shield bubbles. *SuperTuxKart* has become a much more complex game with plenty of play modes, various challenges, scores, championships and other cool tricks other than the usual competition between you and AI players. If you feel like it's still not enough, check out extra content under the Addons section of the game's main menu. There are plenty of extra carts, tracks, arenas and user-created artwork to choose from. You'll never get bored with *SuperTuxKart*!

STRATEGY GAME

Fheroes2

Version: 0.9.7 **Web:** https://github.com/ihhub/fheroes2

Improve your castle to discover extra abilities that will help you win in skirmishes.

The open source project *Fheroes2* is advancing towards its first major release. We decided to play the latest version 0.9.7 and found out that it's already a near-perfect recreation of the original *Heroes of Might and Magic 2* (*HoMM2*). Moreover, *Fheroes2* adds high-resolution support and dozens of UI improvements together with a much improved AI when playing standard campaigns against computer-driven enemies. It looks like *Fheroes2* is actually more playable than the original commercial game that we once had to emulate or run under a virtual machine on Windows.

Very much like *Julius for Caesar III* (see **LXF233**), *Fheroes2* provide a completely reverse-engineered engine with the game's mechanics for *HoMM2*. This means you still need to have the original game copy with all its artwork. However, the *Fheroes2* build comes with the **demo_unix.sh** script, which downloads the demo package of *HoMM2: Succession Wars* and sets things up in such a way that the game instantly works. Effectively, the game copy is abandonware, so it's entirely legal for personal use.

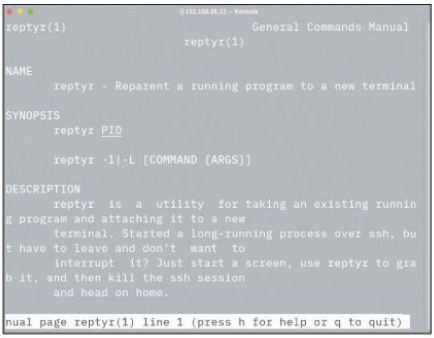
The game itself brings back warm memories of the original *HoMM2*. The standard game drops you at the front of your castle where you have one mounted unit with a couple of sprites. You can explore the surroundings and travel around the map to collect resources (wood and gold, say) and locate your opponents. Don't try to fight anyone during these early days since most other players will outmatch you. Collect as many resources as possible and use your castle to build extra units, then upgrade them.

The game has a complex system of characters that you may develop and improve. Learn new spells, collect wisdom and provide your characters with better armour. Elves, golems and more are just waiting for your word to launch an attack!

TERMINAL RE-PARENT TOOL

Reptyr

Version: 0.8 **Web:** https://github.com/nelhage/reptyr



Process transplantation done swiftly thanks to Reptyr!

This is an amazing Linux trick that enables you to steal a process's terminal output and reconnect it to another terminal session. *Reptyr* can perform 're-pty-ing' of any process in Linux and thus makes it possible to detach a process from one PTY/TTY terminal session and attach it to another. The idea makes use of the 'leading terminal' concept, which is key to Linux. As such, every process in a session has the same controlling terminal, and each terminal belongs to at most one session. If a terminal fails, its child process is killed too, which is best avoided.

Perhaps the best example is running something inside an SSH session and getting disconnected. *Reptyr* enables you to leave the child process running even if its host terminal is killed. That's a fairly common situation: you may want to reboot the server or disconnect from the SSH session and go home without losing things that were running in the terminal. Let's take a look at the example. We'll run a conversion command inside an SSH session, something like this:

```
$ ffmpeg -i input.mp4 output.mkv
```

As long as this is going to take a while, let's reparent this process to another terminal. Log in to the same server via SSH again and run:

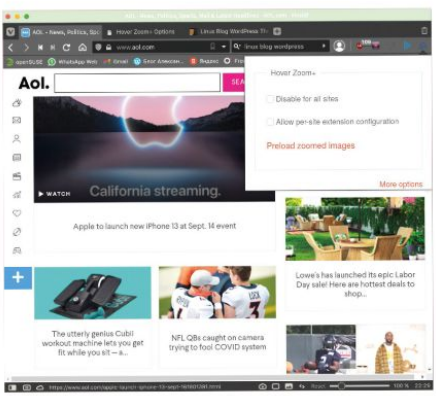
```
$ reptyr $(pgrep ffmpeg)
```

Now that the first terminal is released, you can safely kill the SSH session and stay confident that the process will continue to run on the remote server. *Reptyr's* potential is revealed when used together with terminal multiplexers, such as *Tmux*. It enables the detaching and reattaching of remote processes between sessions, together with their actual output. For improved security we advised you to transplant a process more gently by sending it to background (Ctrl+Z), resuming (`$ bg`) and detaching from the parent terminal (`$ disown <process name>`).

BROWSER EXTENSION

Hover Zoom+

Version: 1.0.180 **Web:** https://github.com/extesy/hoverzoom



Click the extension button to access its settings and optionally preload all images on the current page.

Many of us spend most of our screen time using a web browser. Why not enhance the browsing experience with some open source browser extensions? This time we'll address the issue of content scaling, which seems to be a frequent complaint from high DPI screen users. Even if the UI fonts and other desktop elements scale up correctly, the content rendered by web browsers is a different story.

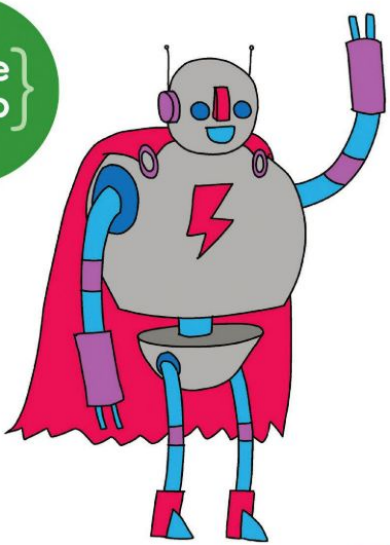
Instead of zooming in and out the entire web page manually, why not bring select images closer on demand? That's what the *Hover Zoom+* extension does. Hover the mouse over any image on a web site and the extension will enlarge the image to its full size, or make it even larger if you increase the scaling factor in the extension's settings. The enlarged image will always fit the page view and never exceed it, therefore the actual scaling factor may be lower for already large images.

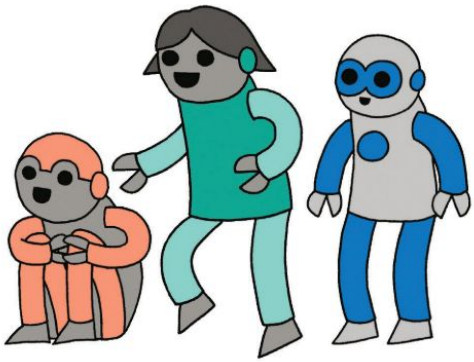
Accessing *Hover Zoom+'s* Options dialog reveals many extra configuration settings. The extension can zoom into images and video, and it can alter audio volume or mute videos, configure delays, show/fade animation duration and more. Customising the *Hover Zoom+* settings allows for a fine-grain per-site control (see the Sites section), including the whitelist mode.

In our tests *Hover Zoom+* worked fine with almost all sites, with a few exceptions. This was because the extension relies on a list of supported domains and site building frameworks, which you can review under the Plugins section. The list covers many news sites and other popular online destinations.

Hover Zoom+ is available for *Firefox*, *Chrome* and *Edge* browsers. Other *Chromium*-based browsers such as *Vivaldi* and *Opera* also support it via their *Chrome* Web Store integration. LXF

{code club}





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**

REST API

Create a RESTful server in Go

Part One! Don't miss next issue, subscribe on page 26!

Learn the theory behind REST as **Mihalis Tsoukalos** explains how to develop a concurrent RESTful server in Go.

OUR EXPERT

Mihalis Tsoukalos is a systems engineer and technical writer. He's the author of Go Systems Programming and Mastering Go. You can reach him at **@mactsouk**.

This month's tutorial covers how to go about developing RESTful servers in Go. The second part will show how to develop command line clients to access this RESTful server. Remember that the single most important task of a RESTful server is the definition of the REST API. With that in mind, let's begin learning about RESTful services.

REST, or REpresentational State Transfer, is an architecture for designing web services. REST isn't tied to any OS or system architecture and isn't a protocol; however, to implement a RESTful service, you need to use a protocol such as HTTP.

Additionally, REST can work with any data format. Usually REST means JSON over HTTP. There are also times where data is exchanged in plain text format, usually when the exchanged data is simple and there's no need for using JSON records. Therefore, when developing RESTful servers in Go, you need to create the appropriate Go structures and perform the necessary marshalling and unmarshalling operations for the exchange of JSON data.

Go for develop

Go is ideal for developing both servers and clients for REST APIs for many reasons. Go works well with JSON data and supports concurrency by design, to name but two. As a result, all RESTful servers are concurrent without needing any extra code because the packages used for serving client requests operate concurrently. However, the principles of concurrent data sharing still apply and you shouldn't share variables carelessly.

The first version of the server is going to use packages of the standard Go library. The second RESTful server is going to use an external Go package that's much better than the default Go package. This first version, which is used of illustrating the ideas behind the development of RESTful services, is going to support the following endpoints:

- **/time:** this endpoint returns the current date and time to the client. It's mainly used for testing that the server and the client can communicate without issues.
- **/insert:** this enables you to insert a new record to the server. As a result, it requires a JSON record as input.

```
func main() {
    arguments := os.Args
    if len(arguments) != 1 {
        PORT = ":" + arguments[1]
    }

    mux := http.NewServeMux()
    s := &http.Server{
        Addr:         PORT,
        Handler:      mux,
        IdleTimeout:  10 * time.Second,
        ReadTimeout:  time.Second,
        WriteTimeout: time.Second,
    }

    mux.Handle("/time", http.HandlerFunc(timeHandler))
    mux.Handle("/insert", http.HandlerFunc(insertHandler))
    mux.Handle("/list", http.HandlerFunc(listHandler))
```

This is the implementation of the main() function of the server.go RESTful server using the default Go router. Each mux.Handle() call associates an endpoint to a handler function.

- **/list:** this last endpoint sends the contents of a map with the data to the client.
- Finally, there's a default handler for all endpoints that don't match an existing one, which in the default Go router is associated with the **/** endpoint.

This simplistic RESTful server accepts all HTTP methods for all its endpoints apart from **/list**, which only works with GET. This is bad practice that we're going to correct later on in this tutorial in the server, which is implemented with **gorilla/mux**. The main reason for doing that is that **gorilla/mux** is better at defining the supported HTTP methods whereas the default Go router requires you to manually use code.

The Go router

Let's implement a RESTful server using the default Go router. The figure (*above*) shows the implementation of the main() function of the RESTful server. First, we create a new router using `http.NewServeMux()`. After that, we define the parameters of the HTTP server using the **http.Server** structure. The next step is about creating the endpoints and their handler functions using `mux.Handle()`. Each call to `mux.Handle()` assigns a handler function to an endpoint. So for **/time** the handler function is called **timeHandler**. It enables multiple endpoints to be served by the same handler function when appropriate, but not the other way around.

QUICK TIP

The official Go web sites are https://golang.org and https://go.dev. You can also read The Go Programming Language by Alan Donovan and Brian Kernighan as well as Mastering Go by yours truly!

```
34 func insertHandler(w http.ResponseWriter, r *http.Request) {
35     if r.Method != http.MethodPost {
36         http.Error(w, "Error:", http.StatusMethodNotAllowed)
37         fmt.Fprintf(w, "%s\n", "Method not allowed!")
38         return
39     }
40
41     d, err := io.ReadAll(r.Body)
42     if err != nil {
43         http.Error(w, "Error:", http.StatusBadRequest)
44         return
45     }
46
47     err = json.Unmarshal(d, &user)
48     if err != nil {
49         log.Println(err)
50         http.Error(w, "Error:", http.StatusBadRequest)
51         return
```

This is the implementation of the /insert endpoint for server.go. The JSON record sent from the HTTP client is converted into a Go structure using a call to json.Unmarshal().

The next section discusses JSON marshalling and unmarshalling, which is required for sending and receiving JSON over computer networks.

The Marshall plan

When working with JSON data over network connections, marshalling and unmarshalling are compulsory processes for exchanging JSON data. Marshalling is the process of converting a Go structure into a JSON record. Unmarshalling is the process of converting a JSON record given as a byte slice into a Go structure. You usually want that when receiving JSON data via computer networks or when loading JSON data from disk files. `json.Marshal()` is used for marshalling and `json.Unmarshal()` is used for unmarshalling.

The Go structure used in this tutorial is the following:

```
type User struct {
        Username string `json:"user"`
        Password string `json:"password"`
}
```

The structure, which is named User, has two fields named Username and Password. The `json:"user"` and `json:"password"` strings found in the definition of the User structure are called structure tags. These tags help you work with JSON records that contain the user and password fields.

The main bug when converting JSON records into Go structures and vice versa is not making public the fields of your Go structures. Go follows a simple rule that states functions, variables, data types, structure fields and so forth that begin with an uppercase letter are public, whereas functions, variables, types and so on that begin with a lowercase letter are private.

Apart from the code that's related to JSON, what's interesting are the implementations of the handler functions. The code for the **/time** function is as follows:

```
func timeHandler(w http.ResponseWriter, r *http.
Request) {
        t := time.Now().Format(time.RFC1123)
        Body := "The current time is: " + t + "\n"
        fmt.Fprintf(w, "%s", Body)
}
```

The aforementioned code ignores the data sent from the client, constructs the date and time string and the body of the response, and then sends it to the client using `fmt.Fprintf()`.

The code for **/insert** as implemented in `insertHandler()` is presented in the screenshot (*left*). Line 35 shows how to check if the HTTP method that was used by the client (`r.Method`) matches the desired HTTP method (`http.MethodPost`). Additionally, we read the client data (`r.Body`) and unmarshall it into a User structure. If the value of the Username field isn't empty, we insert that structure into the DATA map.

The code for **/list** sends the contents of the DATA map to the client – this happens with a `fmt.Fprintf()` call. When data is kept into a slice, there's a different

```
203
204     // Register GET
205     getMux := mux.Methods(http.MethodGet).Subrouter()
206     getMux.HandleFunc("/time", TimeHandler)
207     getMux.HandleFunc("/list", ListHandler)
208
209     // Register DELETE
210     // Delete User
211     deleteMux := mux.Methods(http.MethodDelete).Subrouter()
212     deleteMux.HandleFunc("/delete", DeleteHandler)
213     deleteMux.HandleFunc("/delete/", DeleteHandler)
214     deleteMux.HandleFunc("/delete/{username}", DeleteHandler)
215
216     // Register POST
217     // Add User + Login + Logout
218     postMux := mux.Methods(http.MethodPost).Subrouter()
219     postMux.HandleFunc("/insert", InsertHandler)
220
```

This is the implementation of the main() function for the RESTful server that uses gorilla/mux. The highlighted code implements the subrouters for the HTTP methods.

QUICK TIP

Get the full code for this project on the LXFDVD or head to http://linuxformat.com/archives to download the code pack.

» HTTP STATUS CODES

According to the HTTP protocol you can define the HTTP method of a request, and RESTful services are no exception. POST is used for creating new resources, GET is used for reading (getting) existing resources and PUT is used for updating existing resources. PUT requests should contain the full and updated version of an existing resource. Additionally, the DELETE method is used for deleting existing resources. Finally, PATCH is used for updating existing resources. A PATCH request only contains the modifications to an existing resource.

The most commonly seen HTTP status codes are 200, this means that everything went well and the specified action was executed successfully; 201, which means that the desired resource was created; and 202, which means that the request was accepted and is currently being processed (this is usually used when an action takes too much time to complete). Additionally, 301 indicates that the requested resource has been moved permanently – the new URI should be part of the response. This is rarely used in RESTful services because usually you use API versioning.

Elsewhere, 400 indicates that there was a bad request and that you should change your initial request before sending it again; 401 means that the client attempted to access a protected request without authorisation; and 403 means that the client doesn't have the required permissions for accessing a resource even though the client is property authorised. In UNIX terminology, 403 means that the user doesn't have the required privileges to perform an action.

Finally, 404 means that the resource wasn't found; 405 indicates that the client used a method that isn't permitted by the type of resource; and 500, which means internal server error. Indeed, 500 probably indicates a server failure.

process that should take place – this is going to be illustrated later on.

The curl test

At this point, it's useful to test the operation of the RESTful server, even though there's no dedicated client available. So, in that case we're going to use *curl* to interact with the RESTful server. After making sure that the server is already up and running, we'll test it using the following *curl* commands:

```
$ curl -X POST -H 'Content-Type: application/json' -d '{"user": "mtsouk", "password" : "admin"}' http://localhost:1234/insert
$ curl http://localhost:1234/time
$ curl http://localhost:1234/list
$ curl http://localhost:1234/testingDefault
```

The output of the first *curl* command shows how to use the **/insert** endpoint. The output of the second command checks the operation of the **/time** handler. The third command tests **/list** whereas the final command tests the default handler.

The next section discusses **gorilla/mux**, which is an advanced package for creating web services.

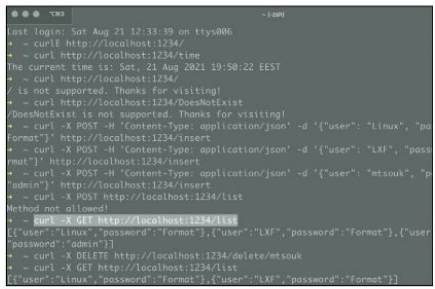
Gorilla muck

This implementation uses the **gorilla/mux** package (**https://github.com/gorilla/mux**), which is a popular and powerful alternative to the default Go router. Although there are many differences between the default Go router and the **gorilla/mux** router, the main difference is that **gorilla/mux** supports multiple conditions when matching a route with a handler function and subrouters. A subrouter is a nested route that's only examined for potential matches if the parent route matches the parameters of the subrouter. The good thing is that the parent route can contain conditions that are common among all paths which are defined under a subrouter. This includes hosts, path prefixes and, as it happens in our case, HTTP request methods. As a result, our subrouters are divided based on the common request method of the endpoints that follow. Not only does this optimise the request matches, but it also makes the structure of the code easier to understand.

The REST API shown here is going to support four endpoints – usually RESTful servers support a much larger number of endpoints. These endpoints are **/time**, **/insert**, **/delete** and **/list**. As before, only **/insert** requires JSON input whereas **/list** returns an array of JSON records. Additionally, **/delete** requires user input in order to know the username that's going to be deleted, if it can be found in the "database".

```
135     if len(username) == 0 {
136         log.Println("username value not set!")
137         rw.WriteHeader(http.StatusNotFound)
138         return
139     }
140
141     // Remove multiple elements if they exist
142     // As the DATA slice changes, we need to call it as a function
143     for {
144         if RemoveElement(username) == false {
145             break
146         }
147     }
148
149     err := writeDATAFILE()
150     if err != nil {
151         fmt.Println(err)
152         return
153     }
```

Figure 4: This is the implementation of the DeleteHandler() handler function that uses the RemoveElement() helper function for deleting all entries that contain the given username from the DATA slice.



This screenshot shows how to interact with the gorillaServer.go using curl, including inserting data, deleting data and viewing existing data.

Because we're using **gorilla/mux**, which is a Go package that's not part of the standard Go library, the development of the program is going to take place under **~/go/src**. More specifically, under **~/go/src/restserver** – you can use any folder you want as long as it's under **~/go/src**.

Data persistency

This time the server stores its data on a text file and reads it each time it starts. In practice, this means that data isn't lost when you restart the server. However, because this is a concurrent server, race conditions can happen when adding or changing data, as multiple Go routines might try to add or change the same records. The easiest solution would have been to use a database for storing data. However, this is beyond the scope of this tutorial.

When the program begins, we read the contents of the data file and put it into a slice. Additionally, each time there's a change in the data, the server should write the contents of the slice back to the file system. This is a naive solution that shouldn't be used in production. The two functions used for data persistency are `readDATAFILE()`, for reading the filename stored in the DATAFILE global variable and putting its data into the DATA slice of structures, and `writeDATAFILE()` for writing the contents of the DATA slice of structures into the path specified by DATAFILE.

The implementation of main()

In this section we're going to discuss the `main()` function – the earlier screenshot (*page 89*) shows the code of `main()`. The most interesting part of it is the use of subrouters, one for each supported HTTP method. Other than this, the code is pretty similar to the one found in **server.go** developed earlier. Additionally, at the beginning of `main()` we call `readDATAFILE()` for reading the existing data from disk. Once again, there's a handler function for everything that's not a match. However, in the case of **gorilla/mux**, the way we specify the default handler function is different than before. Instead of using the `/` path, we provide the `mux.NotFoundHandler = http.HandlerFunc(DefaultHandler)` statement. This means that **DefaultHandler** is going to get executed when there's not a match. Moreover, the parameter of the

» CREATING A DOCKER IMAGE

In recent years it's become common for services to be deployed in Kubernetes as Docker images, so it would be handy to learn how to create a Docker image for the RESTful server developed in this tutorial. The process is easy, provided that you have the appropriate Dockerfile. The screenshot (*below*) shows the contents of the Dockerfile used for creating the Docker image for the RESTful server that uses `gorilla/mux` developed in this tutorial.

The key here is to put your source code under **$GOROOT** (the actual value doesn't matter because this depends on the Docker image) to be able to use Go modules and create the binary executable inside the Docker image that you're using. To make the final Docker image smaller in size, the building of it requires two stages. The first stage is about building the executable. Go binaries are statically linked, so once built they don't require any shared libraries. You then place that executable file to another, smaller Docker image. The second step occurs with the `COPY --from=builder /pro/server /pro/server` command. Finally, remember to make available the TCP port that your server listens to (`EXPOSE 1234`).

After having the appropriate Dockerfile, run `docker build -f Dockerfile -t restlxf .` to create the Docker image. Running `docker images` should show a new Docker image named **restlxf**. You can run the `restlxf` Docker image as `docker run -it -p 1234:1234 restlxf` and interact with it. Just make sure that you're using the correct parameters in the -p flag. In this case, we tell Docker that the server listens to port 1234 internally and externally.

/delete endpoint is given in the URL, is extracted by **gorilla/mux** and is stored as the value of the `username` key.

Finally, imagine that you have a slice of Go structures that represent JSON records that you want to process, as with the **/list** endpoint. Should you process these records one by one? It can be done, but does it look efficient? The good thing is that Go supports the processing of multiple JSON records as streams instead of individual records, which is faster and more efficient. This process is implemented in the `listHandler()` handler function that calls the `SliceToJSON()` function.

This section explains the operation of the **/delete** handler function, which is called `DeleteHandler()`. The most important part of `DeleteHandler()` is the following:

```
vars := mux.Vars(r)
username := vars["username"]
```

The previous code shows how to read all variables and extract the value of the `"username"` key. After that we're free to delete all structures with the Username field equal to the given value. The screenshot (*bottom of facing page*) shows the entire implementation of `DeleteHandler()`. The `RemoveElement()` function is used for deleting one record at a time from the DATA slice because you can't iterate a slice with a for-range loop while deleting elements from it.

The implementation of the handler function for **/time** is the same as before because the functionality is the same. However, in this implementation **/time** only supports the GET HTTP method, which is defined in the subrouters in the `main()` function.

More or less, the Go code for the implementation of the remaining handler functions is almost the same with small changes. The code that changes should have to do with the actual functionality of an endpoint. Put simply, the core functionality of **/delete** and **/insert** is going to be fundamentally different or there's something wrong with the definition of the endpoints!

Testing the server

First, we need to run the RESTful server, which due to the use of Go modules, requires running the next commands inside **~/go/src/restserver**:

```
$ go mod init
$ go mod tidy
$ go run gorillaServer.go
```

The first two commands are required for enabling Go modules and downloading the external dependencies of the code, and should be executed only once. The last command is used for running the RESTful server and should be run each time you want to use the RESTful server. Use `go build` to create a binary executable.

After running the server, we're going to test the RESTful server using *curl*. When sending JSON data to a RESTful server, we need to add `-H 'Content-Type: application/json'` to *curl* to specify that we're going to work using the JSON format. Additionally, the `-d` option is used for passing data to a server and is equivalent to the `--data` option, whereas the `-v` option generates more verbose output if we necessary. The full list of *curl* commands as well as their output can be seen in the screenshot (*top of facing page*). Feel free to experiment with **gorillaServer.go** on your own.

Next month we'll develop a client for the RESTful server implemented here. Until then, experiment and try to create your own RESTful servers. If you want to learn more about the HTTP protocol, you should visit RFC 7231 at **https://datatracker.ietf.org/doc/html/rfc7231**. The website of *curl* is over at **https://curl.se**. LXF

```
5  FROM golang:alpine AS builder
6
7  RUN apk update && apk add --no-cache git
8
9  RUN mkdir $GOPATH/src/server
10 ADD ./gorillaServer.go $GOPATH/src/server
11 WORKDIR $GOPATH/src/server
12 RUN go mod init
13 RUN go mod tidy
14 RUN go mod download
15 RUN mkdir /pro
16 RUN go build -o /pro/server ./gorillaServer.go
17
18 FROM alpine:latest
19
20 RUN mkdir /pro
21 COPY --from=builder /pro/server /pro/server
22 EXPOSE 1234
23 WORKDIR /pro
```

These are the contents of the Dockerfile used for putting gorillaServer. go into a Docker image, compiling it and creating a Docker image named restlxf.

PYTHON

Part One!
Don't miss next issue, subscribe on page 26!

Code a multiplayer shoot-em up game

In the first of a two-part series, **Andrew Smith** takes you though coding a multi-player shooter in good old Python.

OUR EXPERT

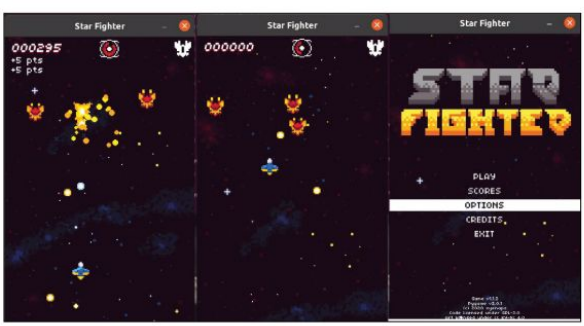
Andrew Smith is a software developer for NHS Digital, has a bachelors degree in software engineering and a master's degree in computer networks.

The merciless forces of Hell have launched a deadly invasion in our region of deep space, and you have been recruited to repel this attack. Mankind is depending on you! Yes, this month we're going to look at *Star Fighter*, a video game created by Francis Michael Tayag. *Star Fighter* has been written as a classic spaceship shoot-em up style arcade game, which is played as a single player taking on computer-generated opponents.

The video game project consists of game music, sound effects, animations, bullets shooting from the spaceship and enemy characters with associated attack patterns, not to mention a fully implemented menu system to configure game options. A player can also enter their name on a scoreboard at the end, which is saved. Many of these features will be covered in this month's instalment. To get started we need a few things: Python, PyGame and the *Star Fighter* source code.

Installation and setup

To install Python, open a terminal window (Ctr+Alt+T) and type `sudo apt-get python3` followed by `sudo apt-get install pip3`. Then install the PyGame module by typing `pip3 install pygame`. To make sure that you're using PyGame version 2.0 (because *Star Fighter* does use a fairly recent version of PyGame), type `python3 -m pip install pygame==2.0.0`. You should now have version 2.0 of PyGame installed.



Star-Fighter in-game shots from its hot Galaxian-style shooting action.

Finally grab a copy of the *Star Fighter* project source code by cloning the GitHub repository. Before typing the following to clone (copy) the GitHub repository, move into a folder on your system that you would like the project to be copied to.

```
git clone http://github.com/zyenapz/Star-Fighter
```

You should now see a folder structure shown in the screenshot (*see facing page, above right*). Navigate into the SOURCE folder where you'll find the main python file for the game, **game.py**. To experience playing the game, type the following:

```
python3 ./game.py
```

Before continuing, take some time to learn what's available to use in the game, such as the menu system which has been implemented, what the game controls are (mainly the cursor keys and Z key for firing) and in general how the game plays.... oh and of course, enjoy blasting the nefarious agents of Hell back to where they came from, before moving on with this tutorial.

Even though **game.py** is the main source file for *Star Fighter*, the game project is broken down into various Python files, which each cover a various aspect of the game. An overview of each file is given below to help you understand how the overall source code is structured:

- **game.py** – Main game program file.
- **defines.py** – Stores values for elements such as screen resolution, game options, directories, pre-defined values and so on.
- **scenes.py** – Deals with different stages of the game program overall, from loading up the game menu, to the Options menu, to the full gameplay to the resultant outcome of the game.
- **widgets.py** – The objects (commonly known as widgets) that make up the options menu selection enabling the player to select values.
- **sprites.py** – Sets the values and properties for each game object in *Star Fighter* including players.
- **spawner.py** – Manages how game characters and objects re-spawn after death/disappearance.
- **muda.py** – Miscellaneous functions such as loading/saving images and data, drawing text and more.

We'll start by looking at what we first see when we load the game program – the Menu system.

As you'll see from running the program, when pressing the up or down cursor keys, the menu option that's selected is highlighted and is executed by pressing Enter. There's also an Options menu that can be selected, which is a submenu.

The Game Menu System is mainly operated by two files: **scenes.py** and **widgets.py**. The best way to understand these two Python files is that **scenes.py** mainly handles the events for each scene of the game program and **widgets.py** handles the contents for each scene, such as the controls the player interacts with to select values for the game.

In an IDE or editor of your choice, open up **scenes.py** and **widgets.py** and then first select **scenes.py** to view. While viewing **scenes.py**, scroll down to or search for `OptionsScene`. `OptionsScene` is a Python class that handles events for the options menu that can be accessed from the main menu.

The code that handles the events for the Options menu is shown below:

```
if self.menu_widget.get_selected_str() == "VIDEO":
  self.P_Prefs.options_scene_selected = 0
  self.manager.go_to(VideoOptionsScene(self.P_Prefs))
elif self.menu_widget.get_selected_str() == "SOUND":
  self.P_Prefs.options_scene_selected = 1
  self.manager.go_to(SoundOptionsScene(self.P_
Prefs))
elif self.menu_widget.get_selected_str() == "GAME":
  self.P_Prefs.options_scene_selected = 2
  self.manager.go_to(GameOptionsScene(self.P_Prefs))
elif self.menu_widget.get_selected_str() ==
"CONTROLS":
  self.P_Prefs.options_scene_selected = 3
  self.manager.go_to(ControlsOptionsScene(self.P_
Prefs))
elif self.menu_widget.get_selected_str() == "BACK":
  self.P_Prefs.options_scene_selected = 0
  self.manager.go_to(TitleScene(self.P_Prefs))
```

From viewing the above source code, you'll learn that each option in the Options menu is treated as a separate "scene" in which a whole list of other options can be presented. You'll see from viewing the source code in **scenes.py**, each "scene" has its own class that manages the events for that scene.

```
andrew@dell-ubuntu-01: ~/PythonProjects/starfight/Star-Fighter/SO...
andrew@dell-ubuntu-01:~/PythonProjects/starfight/Star-Fighter/SOURCE$ ls
data  game.py  game.spec  icon.ico
andrew@dell-ubuntu-01:~/PythonProjects/starfight/Star-Fighter/SOURCE$
```

Here's a terminal listing showing the project structure once cloned and downloaded.

As an example of this, let's look at the Game options. Scroll or search for a class called `GameOptionsScene`.

```
class GameOptionsScene(Scene):
  def __init__(self, P_Prefs):
  # Initialise values
  ....
  def handle_events(self, events):
  for event in events:
    if event.type == pygame.KEYDOWN:
  ....
```

In general, you may find that all classes that are defined in **scenes.py** all look relatively the same with all the events having been dealt with a function (or method) called **handle_events**. The menu system is purely driven by keyboard input events, so there's no event handling for the mouse.

As you look through the code in **handle_events**, you'll notice that a sound effect is triggered with every key press and that the widgets (or input controls) are set to align with whatever option is selected. See below as an example of this:

```
elif event.key == self.P_Prefs.key_down:
  self.sfx_keypress.play()
  # Play sound
  self.menu_widget.select_down()
```

Now let's look at how the widgets (or user controls) relate to this. Select **widgets.py** to view in your IDE or editor and scroll down to or search for `GameOptionsSceneMenuWidget`. In the constructor of `GameOptionsSceneMenuWidget (__init__)`, the position for each of the widgets (controls) is set to align with the options in the menu and to also set what type of control will be used to change the value of the menu option selected.

» OBJECT ORIENTATED PROGRAMMING

Object Orientated Programming (OOP) techniques have been used throughout the *Star Fighter* project. The two main Object Orientated tools that have been used are classes, creating instances of classes (commonly known as an object) and inheritance.

A class can be looked as a template for the object to be created that contains operations known as methods and variables that are known as properties in the world of OOP, which all can help determine how the object will behave once it's created. Let's look at an example that can be found in the source code:

```
class PlayerBullet(pygame.sprite.Sprite):
  def __init__(self, image, position,
velocity):
    super().__init__()
    self.image = image
    self.rect = self.image.get_rect()
    self.rect.centerx = position.x
    self.rect.bottom = position.y self.
position = Vec2(self.rect.centerx, self.
rect.bottom)
    self.velocity = Vec2(velocity.x,
velocity.y)
    self.radius = PLAYER_BULLET_
RADIUS
  ...
```

A class here in the above example is declared as **PlayerBullet**, which is the name of the class itself. Within the definition of the class there exists what's called the class constructor. Here's an example of this:

```
def __init__(self, image, position,
velocity):
  super().__init__()
```

The class constructor is mainly used to initialise property values and perform initial operations when an instance of that class is created. In the class shown in this example, the constructor is used to store an image, set position and velocity values. An instance of this class is declared later on in the source code.

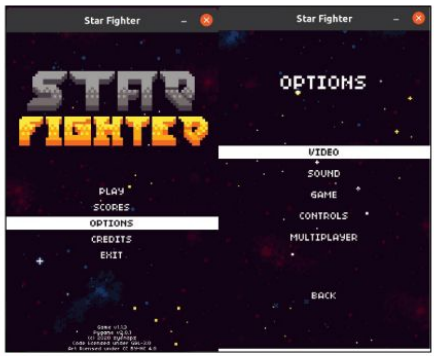
```
  def _attack(self):
```

As can be seen from the above code segment, an instance of the class is created by first calling the class by name and then passing in values to form an object of that class.

```
b = PlayerBullet(self.bullet_image, .....)
```

```
self.ts_hp = TextSelector(self.P_Prefs.hp_pref, HP_
OPTIONS, (x_alignment,self.ts_hp_y),
alignment="CENTER", active=True)
self.ts_canpause = TextSelector(self.P_Prefs.can_pause,
YESNO_OPTIONS, (x_alignment, self.ts_canpause_y),
alignment="CENTER")
```

In this case, as can be seen in the above code segment, a widget called `TextSelector` has been used which has a class declared for it earlier on in **widgets.py**. The values to select on each menu option are declared by `HP_OPTIONS` and `YESNO_OPTIONS`, which can be found in **defines.py**. After initialisation of position and values (in the constructor), there are other functions created that are used to help control the widgets. These are `update`, `draw` (used to draw menu options onto the screen) and various selection options. Before continuing on any further, you may wish to go through the other classes to gain a better understanding how other parts of the game work.



Never overlook the menu system, as it'll be the first thing people see.

Spaceships shooting

Now we'll move on to look at how the shooting from the spaceship is done.

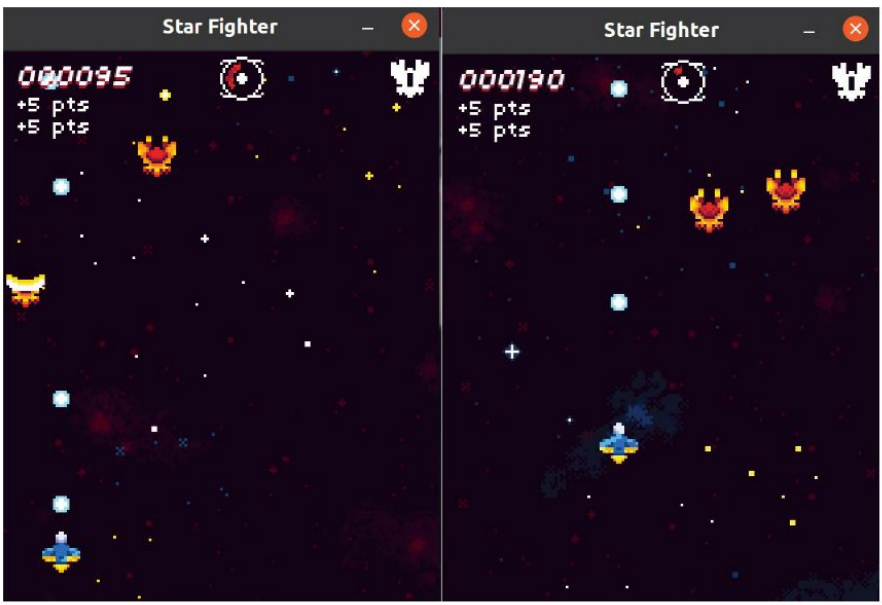
The function that handles shooting from the spaceship is called `_shoot(self, keyspressed)`, this is located in **sprites.py** and is part of the **Player** class that's defined earlier in the file. The key that's used to fire is set up in `P_Prefs` in **game.py** under the name `key_fire`, which is currently set to the Z key. Let's now look at the code for how the firing event is processed:

```
if keyspressed[self.P_Prefs.key_fire]:
  self._play_shoot_sound()
  if self.bullet_increase_timer >= self.bullet_increase_
delay * 2 and self.gun_level == 3:
    self._attack3()
  elif self.bullet_increase_timer >= self.bullet_
increase_delay and self.gun_level >= 2:
    self._attack2()
  else:
    self._attack1()
    self.bullet_increase_timer += self.bullet_increase_
tick
  else:
    self.bullet_increase_timer = 0
```

If the "fire key" is pressed, a sound effect is first played and then an attack sequence is executed depending on the level of gun attained in the game, as well as the timing of the bullet itself. This can be seen from the code segment listed above.

- **self._attack1()** – is an attack sequence that only uses one bullet firing from the spaceship.
- **self._attack2()** – two bullets fire from the spaceship.
- **self._attack3()** – three bullets fire from the spaceship.

The bullet object that shoots from the spaceship is handled by a class called `PlayerBullet`, which on creation of a `PlayerBullet` object instance, takes in the



We focus on creating attacks from your spaceship, so you can shoot at enemy players.

» SPRITE SHEETS

The animations for each of the game's characters and objects are implemented using sprite sheets. These are a method of loading a character animation by just loading one file of multiple images and then loading each image into an array so it can be scrolled through or manipulated and outputted on the screen. They save the trouble of having to load each image individually into memory at a time; by using sprite sheets the images are loaded into memory at once.

If you navigate to the folder **SOURCE\data\img**, you'll find files in that folder such as **hellfighter_sheet.png**, **player_sheet.png** and so on that are sprite sheets used in the game.

A brief example of how the main player sprite sheet is loaded is shown below in **scenes.py**. This example is the same way that other sprite sheets are handled:

```
PLAYER_SPRITESHEET = load_
img("player_sheet.png", IMG_DIR,
SCALE)
  PLAYER_IMGS = {
    "SPAWNING": [
      image_at(PLAYER_SPRITESHEET,
scale_rect(SCALE, [0,144,16,16]), True),
      image_at(PLAYER_SPRITESHEET,
scale_rect(SCALE, [16,144,16,16]), True),
      image_at(PLAYER_SPRITESHEET,
scale_rect(SCALE, [32,144,16,16]), True),
      image_at(PLAYER_SPRITESHEET,
scale_rect(SCALE, [48,144,16,16]), True) ],
```

As can be seen from the code, the sprite sheet is loaded in using a helper function in **muda.py** called `load_img`. The variables passed in can be found in the source file **defines.py**. The images on the sprite sheet are loaded in by a helper function called `image_at`, which is used to grab the image from the sprite sheet into memory.

This code section shows images collected from the sprite sheet that are used when the player is "spawning" in the game. If you read on in the source code file **scenes.py**, then you'll see that images are grabbed from the sprite file for the player who's moving forward, backward, left or right.

following arguments to set up a bullet: image, position and velocity. You'll see that position is a 2D vector (x,y) and velocity is also a 2D vector (x,y).

Enemies shoot back

Staying in the Python script file, **sprites.py**, the enemy shooting first starts with the class called `EnemyBullet`. This class defines the base behaviour and properties for both enemy bullets used in the game.

Again, much like the `PlayerBullet`, the class constructor takes in an image, position value which again is a 2D Vector, velocity which is a 2D Vector. The `EnemyBullet` class also takes in a value for damage that the bullet will do if it hits the player's spaceship.

There are two types of bullet created for the enemy players to use: `EnemyBullet` and `FattyBullet`. Each is defined and implemented with their own class. Even though both are similar in their structure and operation, they're used in slightly different ways even though the `EnemyBullet` class is used as the base behaviour for both enemy characters in the game. The `FattyBullet` class builds on top of `EnemyBullet`. The best way to understand this is that `EnemyBullet` represents the default behaviour and characteristics of an enemy bullet. The class `FattyBullet` adds on additional functionality that `EnemyBullet` does not have.

The values for the variables shown used in the classes such as `FATTY_BULLET_DIRECTION`, `SMALL_BULLET_RADIUS` and `BULLET_DAMAGE`, you'll find in the source code file **defines.py**. When trying to view these it's best to use the IDE's search facility.

An example of this can be seen below in **defines.py**.

```
FATTY_LARGE_BULLET_SPEED = {
  "EASY": 200,
  "MEDIUM": 300,
  "HARD": 400
}
FATTY_SMALL_BULLET_SPEED = {
  "EASY": 125,
  "MEDIUM": 150,
  "HARD": 175
}
FATTY_BULLET_DAMAGE = {
  "EASY": 0.75,
  "MEDIUM": 1,
  "HARD": 1.25
}
```

The values are defined depending on which level the player chooses to play the game at. It's fully possible to learn how each of these variables with associated values affect the game and change the values yourself to improve both how the game plays and the difficulty of the game. From understanding what's been covered in this month's tutorial, you should be able to do any of the following tasks:

- Create menu options.
- Change various game settings to alter the performance and behaviour of the game.
- Add new enemy characters.
- Alter or improve existing animations.

Because there's a lot of source code to go though, you may want to go back over areas of the source code to understand them better before making any modification to the project. See you next month when we'll play with multi-player network mechanics! LXF

It only seems fair to offer the enemy some way to defend themselves.

On the disc

Discover the highlights from this month's packed DVD!

DOWNLOAD YOUR DVD
Get code and DVD images at:
www.linuxformat.com/archives

» START HERE

USING THE LXF DVD

Using Linux for the first time can be very confusing. It'll most likely be unlike anything that you've operated before, especially if you're used to Microsoft Windows or Apple macOS.

Generally our DVDs are designed to be run directly, which is to say that when you first power on your PC (or Mac, see below) it should 'boot' from the DVD – so before Windows or macOS even starts to load – with Linux running directly from the DVD. This trick is known as a Live Disc. It enables you to try out the various versions of Linux without having to install or change anything on your PC. Just remove the DVD, restart your PC and it'll be exactly as you left it.

While many systems will boot from a DVD when it finds one, many will not. See below for the standard process for enabling booting from a DVD on various desktops and laptop PCs.

The alternative option is to locate the ISO file on the DVD and write this to your own USB thumb drive and attempt to run that. We recommend using *Etcher* from **https://balena.io/etcher** that's available for Windows, macOS and Linux. Good luck!

BOOT THE DISC

Many PCs should boot automatically if they're turned on with a disc in the drive. If not, many offer an early Boot Menu accessed by tapping a key while powering up from cold: F9 (HP), F12 (Dell, Lenovo), F8 (Amibios) or F11 (Award BIOS). Alternatively, use the BIOS/UEFI to adjust the boot order to start with the optical drive. Again, this is accessed by tapping a key during power up, usually Del but sometimes F1 or F2.

Some new UEFI PCs require access via Windows: holding Shift select its Restart option. If you're still having problems using the DVD then visit **www.linuxformat.com/dvdsupport**

Mac owners: Hold the C key while powering on your system to boot from the disc.

STUNNING AESTHETICS
MIN SPECS: 2GB RAM, 15GB DISK SPACE

Zorin OS 16 Core

64-bit

Zorin OS has been around since 2009 and has established itself as a key player in the game of user-friendly Linux distributions. It comes in three editions: Lite (a 32-bit flavour that was on the DVD in **LXF279**), Core (this version, which we'll get to) and Pro.

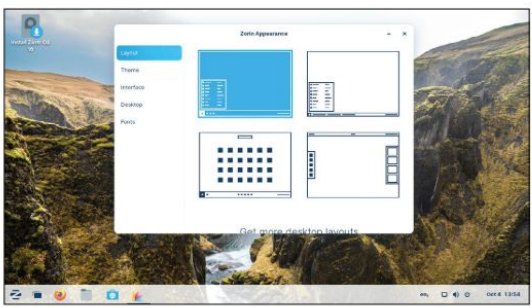
The Pro edition is a paid-for affair (costing £39 plus VAT), and one that's attracted a great deal of following. Its main feature is desktop layouts that can make it look like macOS, Windows 11 and Classic, as well as Ubuntu. This feature makes it much easier for beginners (or indeed experts) to switch away from these platforms. Zorin Pro also includes additional software as well as support with installation. We can't buy it till payday, so we can't tell you precisely what this is, but according to the website it includes a professional-grade creative suite and advanced productivity software. Exciting.

Zorin Core comes with some desktop layouts too (a couple of Windows-ey ones), Touch (that vaguely reminded us of the ill-fated Ubuntu Touch) as well as its own take on Gnome Shell. Zorin Core and Pro are based on Ubuntu 20.04, so they're supported for five years, though you'll probably see a new Zorin release in that time and succumb to upgrading.

Layouts can be switched from the Appearance settings, where you'll also find a couple of other options. Most notably, you'll find a Jelly Mode setting, which is wobbly windows by another name. Like Elementary OS, custom accent colours can be set. We were pleased to see the coder-friendly JetBrains set as the default monospace font, and we must note that in general the default out-of-the-box look and feel is impressive.

As you'd expect there's an awesome app store, which makes it easy to find the software you need. It's integrated with FlatHub too, so you won't be restricted by ageing package versions in the Ubuntu repos. The *Snap* daemon is installed (unlike on Linux Mint) and you can also view software from the SnapCraft store in the *Software* application. We're big fans of Pop!_OS and Elementary OS and since these have adopted Flatpaks we've never once been driven to install via *Snap* there. But it's nice to know the option is there in case FlatHub ever comes up short.

There's also the obligatory Dark theme, plus the option to have it activate automatically at night time. This is in addition to Gnome's native Night Light function, for reducing blue hues in the small hours.



These monochrome diagrams show you the lovely layouts that are included in Zorin.

» IMPORTANT NOTICE!

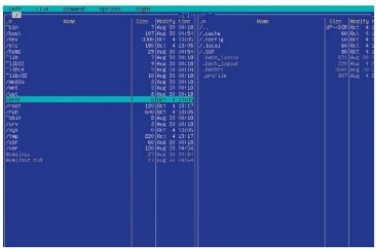
DEFECTIVE DISCS: For basic help on running the disc or in the unlikely event of your *Linux Format* coverdisc being in any way defective, please visit our support site at www.linuxformat.com/dvdsupport. Unfortunately, we're unable to offer advice on using the applications, your hardware or the operating system itself.

SYSADMIN'S DELIGHT
MIN SPECS: N/A

Finnix 123

64-bit

Finnix is a small distro with a big history. It was discontinued in 2000, resumed in 2005 and was one of the first live distros especially geared for system administrators. There was no live booting to a GUI at the turn of the millennium, and neither is there in today's Finnix. That's why the ISO is a positively petite 412MB.



Never underestimate terminal-based file managers.

Sysadmins don't need GUIs anyway. No, they need SSH configuration from the kernel command line, on-hand access to all the filesystem tools, and logging in to a root account by default. Oh, and *Midnight Commander* installed out of the box.

Finnix does all of this and much more. In addition to having most of a standard Debian install (it's based on Bullseye), there are a couple of other handy tools that will surely help any sysadmin in distress. The *wifi-connect* helper utility makes command line Wi-Fi connections possible without having to go anywhere near a **wpa_supplicant.conf** file. And the *Rover* utility is a great curses-based frontend for choosing Dkpg alternatives.

Small doesn't need to mean lacking, and Finnix's diminutive size means that it boots in seconds. Giving you more time to puzzle over how to repair things.

LEISURE APPLIANCES
MIN SPECS: N/A

64-bit

Lakka 3.4 & LibreELEC 10

Since we had space on the disc this month we've added two non-booting distro images as well. You'll find gzipped USB images in the **Lakka/** and **LibreElec/** folders (booting from optical media isn't supported). Write these files out with *Etcher* (*see the Start Here column on the left*), boot them, and then experience Lakka – a complete retrogaming platform, and LibreELEC – the Libre Embedded Linux Entertainment Center, which leverages the *Kodi* media player.

Both of these titles you may recognise from their Raspberry Pi (or other small ARM device) ports, but it turns out that you can run them on run of the mill x86 desktops too. Isn't open source great? Lakka can emulate a huge variety of classic gaming systems thanks to *RetroArch* and *libretro*. Different game systems are implement as separate *libretro* cores, so you get a unified frontend but different emulator configurations are kept separate. So, like *Docker* then, but for eight-bit nostalgia.

LibreELEC aims to be small and fast, Just enough of an OS (JeOS) to run Kodi in fact. Kodi can play your local media, watch streaming services and also connect to a huge number of PVR devices or DVB type tuners. It can be expanded through plugins, but ignore the guides showing you how to connect to pirate streams. Most of those plugins have been banned now. LXF

Lakka's logo is rather pleasing, just like spending hours playing video games.

NEW TO LINUX? START HERE...

Never used a Linux before? Here are some handy resources:
● Read our quick-install guide http://bit.ly/LXFinstall ● Looking for an answer? https://askubuntu.com ● Want to delve more deeply? https://linuxjourney.com

AND MORE!

THE LXF LIBRARY

- **Advanced Bash Scripting Guide**
  Go further with shell scripting.
- **Bash Guide for Beginners**
  Get to grips with the basics of Bash scripting.
- **Bourne Shell Scripting**
  First steps in shell scripting.
- **The Cathedral and the Bazaar**
  Eric S. Raymond's classic text explains the advantages of open development.
- **The Debian Book**
  Essential guide for sysadmins.
- **Dive Into Python**
  Everything you need to know.
- **Introduction to Linux**
  A handy guide full of pointers for new Linux users.
- **Linux Dictionary**
  The A-Z of everything to do with Linux.
- **Linux Kernel in a Nutshell**
  An introduction to the kernel written by master hacker Greg Kroah-Hartman.
- **The Linux System Administrator's Guide**
  Take control of your system.
- **Tools Summary**
  Overview of GNU tools.
- **GNU Emacs Manual**
  Six hundred pages of essential information!
- **Producing Open Source Software**
  Everything you need to know.
- **Programming from the Ground Up**
  Take your first steps.

NEXT MONTH

Take the latest Ubuntu for a test drive and enjoy Gnome 40, a new installer, kernel 5.13 and more!

Next-gen file systems

Secure your files with the best storage. From Btrfs to ext4, we put the latest filesystems on test to find the best for you.

Linux on the road

Autograde Linux is speeding up, so we take a driving tour of the project and explain how you can build your own car-based apps.

Run the Analytical Engine

Kick things super old-school as we emulate the Babbage-designed cog-based Analytical Engine using open source.

Run ChromeOS on everything

Schools are loving the Chrome-based OS, so we explain how to turn your old kit into perfect ChromeOS machines...

Contents of future issues subject to change – we might have run out of petrol.

LINUX FORMAT
The #1 open source mag

Future Publishing Limited,
Quay House, The Ambury, Bath, BA1 1UA
Email linuxformat@futurenet.com

EDITORIAL
Editor Neil Mohr
Windows Insider Jonni Bidwell
Art editor Efrain Hernandez-Mendoza
Operations editor Cliff "Still on Holiday" Hope
Production Ed Ricketts
Group editor in chief Graham Barlow
Senior art editor Jo Gulliver
Editorial contributors
Mats Tage Axelsson, Michael Bedford, Neil Bothwick, Sean Conway, Matthew Hanson, Jon Masters, Nick Peers, Les Pounder, Michael Reed, David Rutland, Andrew Smith, Shashank Sharma, Mihalis Tsoukalos, Alexander Tolstoy
Cover illustration magictorch.com

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
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.
Head of Print Licensing Rachel Shaw

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

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
Chief audience and ecommerce officer Aaron Asadi
MD, tech specialist Keith Walker
Head of art & design Rodney Dive
Commercial finance director Dan Jotcham
Printed by Wyndeham Peterborough, Storey's Bar Road, Peterborough, Cambridgeshire, PE1 5YS
Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU www.marketforce.co.uk
Tel: 0203 787 9001

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier

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

LOVE PAPER

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