GET INSIDE THE KERNEL

Discover how Linux is faster, stable and more secure

» Coding » Graphics » File system » Real-time » Tux

FREE DOWNLOAD POP!_OS 21.04

LINUX FORMAT

The #1 open source mag

PLUS!

- NEXTCLOUD Pi
- TERMINAL BROWSING
- RUN CLASSIC DISTROS

CELEBRATE 30 YEARS OF LINUX!

We explore the development that took Linux from a zero to a kernel hero!

67 pages of tutorials & features

Create an advanced virtual network with VirtualBox

Design documents like a pro with Scribus

Build a custom digital signage system with Xibo

APP SECURITY
Auto-scan and catalogue code

TILE GAME CODING
Everything you need to make a puzzle slider

LXF September 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

We're celebrating the triumph of 30 years of the Linux kernel, so what's been your highlight over that period?

Jonni Bidwell
Adobe Flash was born kicking and screaming (and otherwise animating colourful objects) into this world in 1996. This makes it just a little younger than Linux, and definitely not a thing the past three decades should be proud of. Nonetheless, we should all rejoice in its overdue demise.

Les Pounder
My highlight of the past 30 years is somewhat recent. Oggcamp is a FOSS event that I've been a part of since 2009. In that time Oggcamp has generated new communities and ideas. More importantly, friendships have been made, and that's the true measure of Oggcamp's success.

Mayank Sharma
Three decades is a long time, and though you weren't specific I'm gonna skip getting married, and the birth of our mini human overlord from the highlights, and instead doff my hat to virtualisation, both on the desktop, and in the form of app containerisation thanks to Docker.

Shashank Sharma
The release of Knoppix in 2000 was a watershed moment – it turned things around for Linux. Gone were the days of having to suffer through the horrors of installation to try something new. Now, you could just pop in the CD and immerse yourself in a whole new experience.

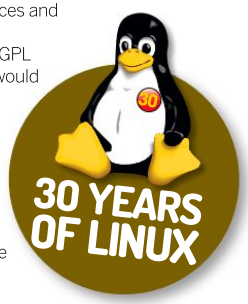
Alexander Tolstoy
Pick out a single highlight from the 30 years? Doesn't seem possible, even for a seasoned Linux user and a happy father of four like me! But one of my memorable technical highlights was going totally Windows-free in the mid-2000s thanks to Linux a mature-enough replacement.

Tux turns 30

In the sage words of Kool & The Gang, "Celebrate good times, come on." We are, of course, talking about me having edited over 100 issues of *Linux Format* <whispers from Jonni> but also that 30 years ago almost to the day a young Linus Torvalds announced on **comp.os.minix** that he'd started work writing a kernel (nothing big or professional like GNU) running on his i386 PC. The rest, as they say, is history.

For this special occasion we're taking time out to look back over those 30 years of Linux development (for which *LXF* itself has been around for 21 years) and ask how did that happen? We follow kernel growth over the years, reveal how distros expanded out of those early days to foster an open source OS ecosystem, and chart how an industry expanded alongside all of this to conquer the computing world, to the point where even Microsoft now embraces and apparently loves Linux.

If it wasn't for young Linus adopting the GPL those decades ago, I don't think the world would be quite the same. Having such a strong copyleft licence at the heart of the Linux world has helped cement the developer communities that enabled the Linux kernel to be used and reused in ever-growing fields. Let's all remember and celebrate the hard work, open sharing and the communities across the globe that have helped Tux spread around the globe. Enjoy!

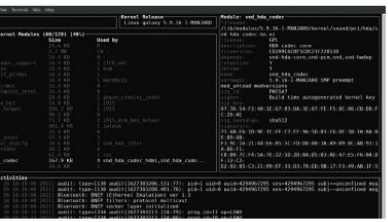


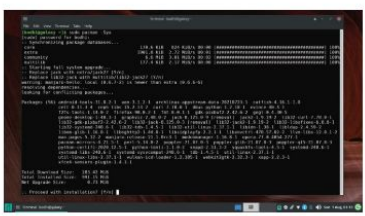
Neil

Neil Mohr Editor
neil.mohr@futurenet.com

Contents

REVIEWS

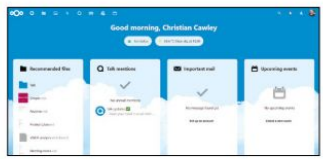




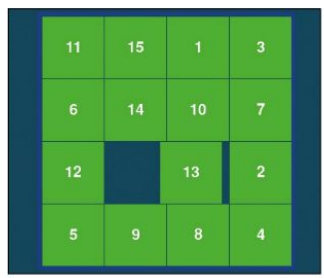
ROUNDUP

IN-DEPTH

Pi USER



CODING ACADEMY

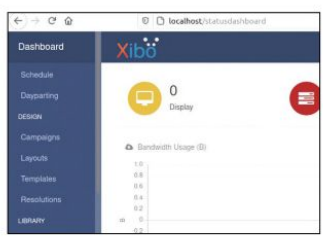


REGULARS AT A GLANCE

ON YOUR FREE DVD
- Pop!_OS 21.04
- Classic distros

See page 96

TUTORIALS



Newsdesk

THIS ISSUE: Steam Deck unveiled » Audacity maintainer threat » Foundation's gaming angle » Funding GIMP » Stockfish legal action

GAMING

Valve announces the Linux-powered Steam Deck

Upcoming handheld PC will use SteamOS, which is now based on Arch, and it could have big benefits for Linux gaming.

Valve has revealed its new handheld gaming device, the Steam Deck (**https://bit.ly/lxf280steamdeck**). While its specs means it should comfortably outperform Nintendo's Switch, the key takeaway is that the Steam Deck runs on SteamOS 3.0, which is now based on Arch with the KDE desktop environment.

Since the announcement of the Steam Deck, there's been much talk on the openness of the hardware and what it could mean for Linux gaming. When we say 'openness' we should be careful. Steam, after all, is a proprietary store that uses DRM. It's also yet to be seen if SteamOS 3.0 on the Steam Deck will allow other games launchers and stores to be installed.

Gabe Newell, CEO of Valve, has been clear that it should be easy to replace SteamOS with another operating system, be it other Linux distros or even Windows 11. In a reply to a question emailed to him (**https://bit.ly/lxf280gaben**), about whether or not the Steam Deck will be locked down, or if other OSes can be installed, he replied that "Yep. It's a PC. You can do whatever you want." He even suggests other manufacturers can build Deck clones.

This is in marked contrast to most other games consoles that won't let you easily install other operating systems. If the Steam Deck becomes a success, then it could also encourage games developers to port their games to Linux, something that has slowed of late.

Valve has also been working on its *Proton* feature, which is a compatibility layer (based on *Wine*) that integrates with Steam (and SteamOS) and enables you to play Windows games in Linux. The Steam Deck, at least initially, will be relying on *Proton* to ensure that people can play their large Steam libraries on the device. This means Valve will continue to improve *Proton* and add features, and that's great news for all Linux gamers.

The new Valve Steam Deck runs on Linux and could bring more games to the operating system. We hope.

However, Feral Interactive has just announced that it has shelved plans to port *Total War Saga: Troy* to Linux, explaining that, "there is generally less demand for native titles since Valve's launch of *Proton*" (see **https://bit.ly/lxf280feraltweet**).

VALVE'S DOMINO EFFECT IN ACTION

If the Steam Deck becomes a success, then it could also encourage games developers to port their games to Linux.

This can be seen as a positive, if *Proton* ends up making Windows games run as well as a native Linux port would. However, it could mean fewer native Linux games, and while you can use *Proton* with non-Steam games, you still have to have a Steam account and add those games manually. While the Steam Deck bodes well for Linux gaming, there's a degree of concern, too.

SOFTWARE

Audacity drama continues

Anger over Audacity's use of telemetry reaches boiling point as 4chan gets involved.

Back in **LXF278** we reported on how *Audacity*, the open-source audio editor, was acquired by the Muse Group. Although promising to keep *Audacity* free and open, the company added optional telemetry to the application which would upload certain data (which is detailed in the *Audacity* sub-Reddit at **https://bit.ly/lxf280audacityreddit**) to third-parties, including Google and Yandex.

This prompted a fierce backlash, and while the company swiftly backtracked and apologised (**https://bit.ly/lxf280audacityapol**), the damage was done. It didn't help that there was also a (now dropped) change to the privacy policy to prevent people under 13 from using the software, which many felt breached the GNU General Public Licence 2, which *Audacity* is published under. Fifty-plus forks of *Audacity* were then created, with one of the most popular being *Tenacity* (**https://bit.ly/lxf280tenacity**).

When the fork was created by a programmer going by the name "cookiengineer", a poll was set up to try and find a new name. Unfortunately, this poll was spotted by users of 4chan, who then engineered to have the name Sneedacity (a reference to a Simpsons joke) win. This led cookiengineer, who felt the poll was manipulated, maliciously to remove it and go with *Tenacity*.

While this should have been the end of it, 4chan users allegedly decided instead to harass cookiengineer which, the programmer claims, included finding their real name and address and threatening him. This led to cookiengineer announcing (**https://bit.ly/3idhUPd**) that they're stepping down from the *Tenacity* project.

The controversy surrounding Audacity has taken a rather dramatic turn.

GAMING

Linux Foundation in game engine push

The foundation teams up with the likes of Adobe and Red Hat to design a 3D game and simulation engine.

Alongside the announcement of the Linux-powered Steam Deck (*see opposite page*), gaming on Linux is taking another exciting step forward with the Linux Foundation joining with other institutions, such as Adobe, Intel, Huawei and Red Hat to create the Open 3D Foundation (**https://bit.ly/lxf280open3d**), which will support open source projects that focus on 3D graphics, rendering and development.

The first such project is the Open 3D Engine (O3DE), which is based on the Amazon Lumberyard engine, and will be made available under the Apache 2.0 licence. This will hopefully make developing games now more accessible and affordable, as devs won't need to pay to licence it. Devs using the engine will be supported via "an open source community through forums, code repositories, and developer events." Features included in O3DE will be "a new multi-threaded photorealistic renderer, an extensible 3D content editor, a data-driven character animation system, and a node-based visual scripting tool," and developers can author code in C++, LUA and Python. Meanwhile, the Open 3D Foundation will have a governing board alongside a Technical Steering Committee.

As Chris Aniszczyk, CTO of Linux Foundation explains in the announcement (**https://bit.ly/lxf280open3dannouncement**), "the new Open 3D Foundation finally gives gaming and engine developers an opportunity to influence the direction of a major AAA-class 3D engine that is sustained for the long term by a worldwide open source community." You can find out more about O3DE at **https://o3de.org**.

OPINION

UPSTREAM FIRST

Gustavo Padovan
Kernel team lead at Collabora, helping companies connect with the mainline community.

"It's thrilling to see Linux reach its 30th anniversary. From the famous "just a hobby, won't be big and professional" Linus' email to today, a lot has happened. Linux is everywhere... and on Mars!

I believe that Linux will continue to evolve, adding countless new deployments for the next 30 years. Machines running Linux are expected to be functional for years, even decades, which means we need to provide kernel updates with new features and security fixes for a long time.

To make this future possible, companies have to engage more with the community, submitting their code to the mainline kernel as fast as possible, in an upstream-first approach. The Linux testing and Continuous Integration(CI) capabilities also need to keep growing. The KernelCI project, which is bringing key industry players together to build a comprehensive CI for the kernel, is a great example of such an effort.

The closer to upstream a company works, the easier and cheaper it is to update devices to newer kernels. With an upstream-first approach, they benefit from the mainline community support and all the CI that we, as a community, are putting in place."

OPINION

FIRST STEPS

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

I downloaded six floppies over my 14,400 baud modem (well, my employer's modem to be more accurate). The downloads took about a quarter of an hour. Then I had to write the floppies out using *rawrite.exe* under Windows.

I shut down Windows and rebooted the PC. Insert the first floppy... and the second... and so on. More time passes. Finally, ready to reboot again. Something about Bogomips flashes past, and then there's a login prompt!

And it was an amazing feeling. A pre-V1 Linux kernel in, I think, 1992. I could even set a task running then switch to another virtual console and do something else at the same time. Windows couldn't do that. This is the future of computing.

Put another way: how I completely misunderstood that Microsoft was a very competent marketing company.

Thus began my journey with Linux. For the past 20 years, it's been my source of income. It doesn't bear much resemblance to those days with half a dozen floppies, but it's still fun and it's still free. Thanks Linus. The first 30 years were pretty good. Here's to the next 30.

SOFTWARE

GIMP seeks funding

Open source image editor is looking to help fund some of its contributors.

GIMP is a popular image editing tool that's looking to help support its contributors.

One of the biggest challenges with open source software is how to retain talented developers. These projects are made by a passionate community and involves a lot of unpaid work, which means developers have to find other means to make money – and this sometimes means they can no longer continue with the projects.

GIMP, the popular photo editing software, is no stranger to this, and the project has decided to highlight crowdfunding efforts by some of its contributors. Both Øyvind Kolås, a GEGL maintainer who's been contributing to *GIMP* since 2004, and the ZeMarmot Libre Art project both have Patreon accounts (**www.patreon.com/pippin** and **www.patreon.com/zemarmot**, respectively) where you can pledge payments to help fund them, with *GIMP* saying that "you can consider these crowdfundings as 'official' as can be and completely endorsed by the *GIMP* project." Hopefully this move will help bring greater attention to the crowdfunding efforts of *GIMP* contributors and help them make a living while working on the project.

LICENCES

Checkmate, GPL abusers!

Chess engine Stockfish files lawsuit against ChessBase over GPL 3 violations.

Throughout history, chess has seen fierce rivalries play out, but perhaps none like this. The open source *Stockfish* chess engine has filed a lawsuit against ChessBase, accusing it of violating the GPL 3 licence *Stockfish* is shared under, with ChessBase's *Fat Fritz 2* chess engine. This, the *Stockfish* team claim, is a derivative of *Stockfish*, but with some new lines of code and proprietary features.

In an announcement regarding the lawsuit (**https://bit.ly/lxf280stockfish**), the *Stockfish* team state that "ChessBase repeatedly violated central obligations of the GPL, which ensures that the user of the software is informed of their rights."

So far, the team's moves have resulted in the *Fat Fritz 2* DVD being recalled and halted sales of ChessBase's *Houdini 6* product, which again is a derivative of *Stockfish*. Stockfish has now terminated its GPL licence with ChessBase, but the company is seemingly still selling its products. The lawsuit, Stockfish claims, will hopefully enforce the licence termination and stop ChessBase from selling products based on the chess engine.

SCHOLARSHIPS

Linux Foundation Training Scholarship winners announced

500 scholarships announced for this year – the highest yet.

Since 2010, when the Linux Foundation Training Scholarships (**https://bit.ly/lxf280lift**) were launched, over 1,100 scholarships have been awarded. In 2021 that number has grown significantly, with 500 being granted for this year alone.

According to Linux Foundation Executive Director Jim Zemlin, this year's high number of scholarships is due to the Covid-19 pandemic, and are aimed at helping people affected by it. "The level of talent and potential evident amongst this year's winners is staggering," Zemlin said. These include a 16 year old from India who teaches tech to younger children and a Kenyan developer who makes open-source software for local businesses and institutions.

People worldwide benefit from the Training Scholarships.

Distro watch

What's down the side of the free software sofa?

SIDUCTION 21.2.0

This distro, based on Debian's "unstable" branch, and with a choice of Cinnamon, KDE Plasma, LXDE, LXQt and Xfce desktops, has a new version out to coincide with the 10th anniversary of the project. An improved manual and updated desktop environments are the highlights of this new release, with Plasma being the main focus. The manual is currently only available in German, but work is ongoing to translate it into English. To find out more, check out the release notes at: **https://bit.ly/lxf280siduction**.

siduction's 10th birthday has been marked with a new version.

4MLINUX 37.0

A new release of this distro, which focuses on the 'four Ms' (maintenance, multimedia, mini-server and mystery) brings updates aplenty to many of the packages included, such as *LibreOffice 7.1.5*, *GIMP 2.10.24* and *Firefox 90.0.2*. The Linux kernel has also been updated to 5.10.47. 4MLinux now also uses its own servers for updating the *ClamAV* virus database, according to the release announcement at **https://bit.ly/lxf2804mlinux**.

4MLinux receives a new stable release that updates many of its most popular packages.

FREESPIRE 7.7

A new update to this Ubuntu-based distro is, according to the release announcement (**https://bit.ly/lxf280freespire**) an "entirely new direction for our distribution products, Freespire, Linspire and Xandros, by incorporating a cloud app approach." For Freespire, that means you can choose which web apps you want to install, and if you'd rather stick with the traditional way of doing things, it's still a full-featured desktop operating system as well, which uses the Xfce desktop.

Despite being a point release, Freespire 7.7 brings some big changes.

OPNSENSE 21.7

At the time of writing, OPNsense, a specialised operating system for firewalls and routers and based on HardenedBSD, has a new version out. It's been described by its developers as "one of the largest iterations of code changes in our recent history," and will be the last release to use HardenedBSD 12.1 before it switches to FreeBSD 13. The installer now supports ZFS installations, and firmware updates have been partially redesigned, among many other changes. Check them all out at **https://bit.ly/lxf280opnsense**.

OPNsense is an open-source firewall that puts user-friendliness at the forefront.

OPINION

IMPROVING INCLUSIVITY

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

Over the past decade, many of us in open source have tried to improve inclusiveness and diversity in our communities. STEM programmes in schools, training programmes at corporations, and efforts within universities have all improved diversity.

The barrier to entry for first timers can be very high and often off-putting. Currently, we aren't always well set up to make new contributors successful when they try to engage. For example, I have seen project maintainers dismiss pull requests or bug reports if they're not the same as those submitted by long-time contributors.

We have to consider how we make the process easier, and offer assistance to those getting in touch for the first time. This covers all areas – from submitting bugs and contributing code through to getting help with documentation.

For these developers, their opinion of open source will be tied to how we help them contribute and whether they feel like they belong. We have to be willing to step outside our own comfort zones and consider new ways of working. This will help new contributors get involved, get started, and contribute to the community over time.

OPINION

ACORNS TO OAKS

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

“This month, we’re celebrating the 30th anniversary of Linux. On 25 August, 1991, Linus Torvalds posted his now famous introductory message to the Usenet newsgroup comp.os.minix: “I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu)”. If only he had known how impactful his work would be.

Today, Linux is the dominant operating system of the modern cloud-based world, but it wasn’t always preordained that this would be the case. Younger readers probably don’t recall a time when Microsoft publicly called Linux a cancer and actively fought it (as contrasted by their contemporary friendly embrace and adoption).

There was a time when Linux was seen as the biggest threat to many traditional companies who would later embrace it.

Thirty years is truly an impressive amount of time for any project. I remember downloading my first full (not “single” disk) version of Linux in 1996 at the age of 14. It required about 200 floppy disks and took several weeks to download Slackware 96. Since then, I’ve gone on to make my livelihood out of working with Linux and open source software. Linux has been good to so many of us. Cheers to Tux!”

KERNEL WATCH

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

Linus Torvalds has announced several Release Candidate (RC) kernels for what will become 5.14 by the time you read this. The 5.14-rc4 kernel (and later) include an important fix to the handling of pipes that’s required to properly fix a regression that had affected Android.

The regression occurred because applications were assuming certain semantic behaviour not part of the specification, but instead part of the existing implementation. In such cases, Linux will always aim not to break userspace at almost all costs, even if applications are using an interface incorrectly. The only time this isn’t true is when a breakage is so isolated that it’s known not to impact anyone, or when a security vulnerability necessitates some kind of unavoidable impact upon userspace. This is why Linus regularly asks developers to test his RC releases. He wants to know about this kind of breakage in time to fix it before it goes into lots of Linux distributions.

There were a number of security issues over the past month. In a moment, we’ll take a look at one of them in more depth. We’ll also draw attention to a blog post from Kees Cook in which he walks through the need for greater investment into improving upstream: **https://security.googleblog.com/2021/08/linux-kernel-security-done-right.html**.

Sequoia

Recently, the folks at Qualys reached out to the Linux kernel community to let everyone know that they’d discovered a local root exploit vulnerability that had been lurking in plain sight since 2014. The vulnerability allows an untrusted local user to gain root by exploiting an erroneous numeric conversion from size_t (a 64-bit signed value on most platforms) to an int (a 32-bit signed value on the same platforms). Since this is a critical security issue, fixes were coordinated among the many impacted vendors and released simultaneously.

The detail of the vulnerability makes for interesting reading. Essentially, the team rely on the size_t-to-int bug in the kernel’s seqfile mechanism (hence the reason for the brandname “Sequoia”). Within the kernel’s **/proc procfs** “pseudo” filesystem, entries aren’t backed by disk storage but instead are created as they are read. Among the many users of procfs is the **/proc/self/mountinfo** file that displays mounted filesystems. When a suitably ridiculous path is created (over 1GB in the filename path) and mounted by a user, reads of this file to ascertain mount info will result in corruption of kernel memory.

An attacker can exploit this corruption, since the kernel will write the text `//deleted` into a memory location that can be calculated. As a result, it’s possible to arrange to load an eBPF program that will be subtly corrupted by the kernel so that the normal security checks are no longer applied. The attacker has just enough control over the corrupted BPF program to overwrite the location of the `/sbin/modprobe` command the kernel will run when detecting new hardware with a different program. That program is run with root privileges next time a hardware event of any kind occurs.

» ONGOING DEVELOPMENT

Mark Rutland posted a patch correcting the kernel’s use of thread_info flags that previously weren’t guaranteed to be used in a safe manner by shared reader/writers. Matthew Wilcox updated his “Memory folios” patches to better support filesystem data stored in the Linux “page cache” by enabling large files to span multiple pages without wasting additional page table entries.

Meanwhile, planning is underway for the upcoming (virtual) Linux Plumbers Conference, taking place at the end of September (20-24). This is a key (community-organised) event in the annual Linux calendar because it’s typically also co-located with the Kernel Summit and draws attendees working on all of the lowest level “plumbing” in the modern Linux software stack.

With the event being virtual this year, it should be more accessible than ever. Now would be a great time to bookmark **https://linuxplumbersconf.org** and watch out for updates.

Answers

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

Neil Bothwick
Has answered 30 problems for 30 years of Linux!

THE BASICS

Q Is a swap partition necessary when installing a distro these days?

A Usually not, but some software prefers to have swap available and it doesn't hurt to give a small amount of disk space for one. The old rule of double the amount of RAM is excessive though, and may actually harm performance. The same amount of RAM is usually more than adequate for modern distros.

Q What are these man pages people keep telling me to read and how do I access them?

A Man, or manual, pages are the documentation supplied with many commands. They're read with the *man* command, so `man man` gives the man page of the `man` command, in true recursive tradition.

Man pages are meant to be read in a terminal – you can press `/` to search them – but there are also online alternative, such as those at **www.mankier.com**, which are often easier to read, and indexed for your convenience, too.

Q I often see comments like "check the logs" or "does syslog tell you anything". What is syslog and what logs do they means?

A The system log was traditionally kept in **/var/log**, often **/var/log/messages**, and most programs put information in here. With *systemd*, the syslog has been replaced with the journal, which can be read with *journalctl*.

Q I have these directories called /proc and /sys that contain a lot of files, some very large. Can I delete anything to free up space?

A These are virtual filesystems that don't use any disk space. Rather, they're a way of examining the internals of the system using the "everything is a file" mantra. You can find process information, disk and memory usage and even a complete memory dump in here.

IT'S DEAD, JIM

Q I get random crashes. There's no real pattern but it seems to happen more when the computer is loaded.

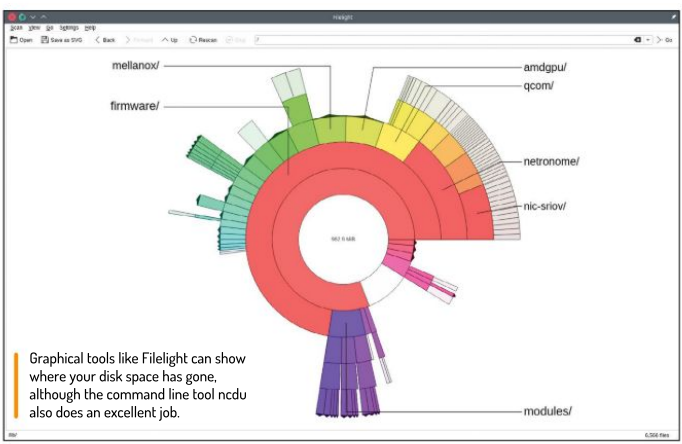
A Check for faulty memory. Boot from an *LXF* DVD and select the *memtest* option. Let it run for at least a couple of passes, preferably overnight, to be sure that your memory is okay. If the memory passes and this is a desktop computer, try a new power supply because failing old ones can cause random crashes.

Q I can't run anything properly. Programs either won't start or behave erratically, and nothing remembers its settings.

A You may have run out of disk space, either on your home or root partition; programs can behave strangely when they cannot write to **/var**. Check with the `df` command. If you're using the ext4 filesystem, it's possible that you've run out of inodes rather than absolute space. Check with `df -i`. In either case, if a filesystem shows 100 per cent usage, delete or move some unimportant files.

Q How to I kill a program that's not responding to Ctrl+C?

A First of all, give it time. Ctrl+C sends a TERM signal, which asks the program to end its process and stop cleanly. If that doesn't work you can send a KILL signal with `pkill -TERM programname`. If there's more than one program with the same name, use *pgrep* to find its PID and then use `kill -TERM PID`. You can also send kill commands from *top*.



Graphical tools like Filelight can show where your disk space has gone, although the command line tool ncdu also does an excellent job.

Q How do I close a window of a program that's either crashed or locked up?

A Run *xkill* – note that you may have to do this from a terminal. The mouse cursor changes to a skull and crossbones. Then click the window you wish to get rid of and its program will be terminated. There are no second chances, so make sure to only click the window you want to kill. Bitter experience has taught us that an over-sensitive touchpad could cause problems.

Q My computer has locked up completely. Ctrl+Alt+Del has no effect. What can I do?

A The "Magic SysReq" key combination helps here. Hold down the Alt and SysReq (sometimes marked PrtScr) keys then press and release in sequence: R, I, E,

S, U, B. These keys tell the kernel to kill all running tasks, sync the filesystems, mount them read-only and then reboot. It's a lot friendlier than killing the power.

Q Sometimes, my computer is so locked up that even the Magic SysReq option doesn't work. I can pull the plug on a desktop, but how to I force a laptop to power down?

A The power button is usually mapped to an ACPI rule to suspend the computer, which won't happen if it's crashed. However, if you hold in down for more than five seconds, it will force a hard power down.

ALL IS NOT WELL

Q I'm trying to start a program from the desktop menu, but nothing happens; no message, no errors and definitely no program.

A Try running the program from a terminal. The command name is usually the same as the name in the menu, but all lower case. If in doubt, use tab completion. It'll probably still fail, but you should get a clue as to why in the terminal.

Q My computer is running slowly lately. I don't think that I have changed anything, but the desktop is noticeably less responsive.

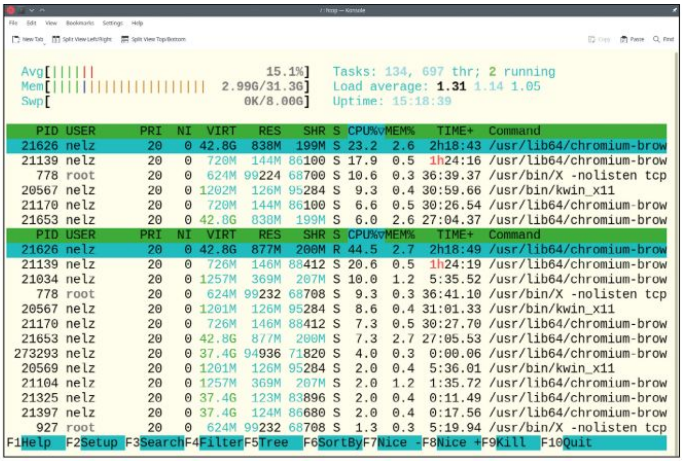
A You may have something running in the background that's hogging system resources. Run *top* in a terminal to see running tasks that by default are sorted by CPU usage. Press M to sort on memory usage or use the < and > keys to move between columns to sort on. If the culprit is your web browser then you may be able to close only the tab that's hitting the system.

Q The lights on my modem are flashing even when I'm not using the internet. I'm concerned that something could be downloading in the background.

A Some activity is to be expected: keeping the connection up or maybe checking mail in the background. To see what's using your bandwidth, install and run *nethogs*, which works much like *top* for network use.

Q I'm running out of disk space, but don't know what is using it all.

A There are a number of disk usage programs; Gnome's *Disk Usage Analyzer*, KDE's *Filelight* or the terminal command *ncdu*. You'll need to be root to check anything but your **/home** directory, and use the **-x** option with *ncdu* to avoid straying into other filesystems.



If the standard top command is too plain for you, try htop, an enhanced alternative with extra features and colours.

Q I tried some of your suggestions, but all I get is "command not found" – why?

A Not all of the commands we suggest are installed by default on all distros, so you'll need to fire up your package manager to install the relevant package.

Q How do I know which package to install to obtain a particular program? I'm using Ubuntu.

A Often you can search for the package name in the package manager, but sometimes a package contains multiple commands so the name isn't a help. You can search **http://packages.debian.org** to see which package contains a specific program or file. This should work for any distro that's based on Debian, including Ubuntu and its spin-offs.

SYSTEM SHENANIGANS

Q I created a new user in Ubuntu and most things are fine, but the new user is unable to run system commands with *sudo*.

A *Sudo* is controlled by the **/etc/sudoers** file, which says who can do what. Normal users can do nothing. Your original user can because they're a member of the *sudo* group. Add your new user to the *sudo* group and *sudo* should work just fine. Other distros may use a different group – either admin or wheel.

Q It seems that certain operations or hardware need the user to be a member of a particular group. How do I know which groups a user belongs to?

A There are a couple of commands that will do this. Running *groups* does exactly what you would expect: it lists your groups. The `id` command does the same, but gives a little more information. Either command can be run with the name of a user, or will give details for the current user if none is given. Add a user to a group with `sudo gpasswd --add username groupname`. The command has to be run as root. However, the user won't obtain the permissions of the group membership until they log out and log back in.

Q I'm not sure everything is starting up correctly. How do I check my system's status using systemd?

A You can do this with systemd's general-purpose `systemctl` command: `systemctl list-units --failed` will show units that failed to start. The `list-units` option normally shows running units, but adding `--failed` means it shows just what you need.

Q Do I need a firewall with Linux, especially because my router already has one?

A Linux doesn't need outgoing firewalls to the same extent as Windows because there's not the proliferation of malware. For incoming connections, it depends on whether you've opened up ports on your router. If you have then you certainly need the router firewall. One on the computer can give a second line of defence, and more information on any intrusion attempts.

Q I'm running a script from *cron*, but it's not working. It works fine when I run it from a shell.

A *Cron* runs scripts in a limited environment, without your full profile. In particular, $PATH is much shorter. As a

result, you should call all commands with absolute paths when running anything from *cron*.

Q How do I track down duplicate files on my disk? I have multiple copies of my photos and music files.

A The `fdupes` command will do this. Run something like `fdupes -r ~ >dupes.txt` to create a list of all duplicate files in your home directory. This command can take a while to run, so save the contents to a file for later examination.

Q I'm not sure that the changes I made to /etc/default/grub have had the desired effect. How can I see exactly what options I'm booting with?

A You can press e (for edit) at the GRUB menu to see the options for the selected entry. Press return to boot with them. If you have already booted then the file **/proc/cmdline** contains the full kernel invocation, including any initramfs inclusions.

THAT'S USEFUL

Q I have a text file containing columns of information, and I want to extract just the second and fourth columns. Can I do this easily?

A You can, using a small fraction of the power of awk, like this:

```
$ awk '{print $2, $4}' textfile
```

You can do much more, such as only use rows that contain a given pattern, or use something other than whitespace as the columns delimiter. The man page provides a lot more detail.

Q I have a directory full of images that I want to resize, I could do this in *GIMP*, of course, but there are many hundreds of them and there's no way I want to tackle them individually! There must be a quicker way.

A There is – you can use the convert command from *imagemagick* to batch resize images, For example:

```
$ for PIC in *.png; do convert -resize 50% $PIC; done
```

will resize all PNG images in the current directory to half the size. Alternatively, you could use `-resize 800x600` to shrink all images to fit that size.

Q My USB scanner doesn't appear as an option in *GIMP* or *XSane*, but it works with Windows.

A This is usually a permissions problem. The terminal command `sane-find-scanner` should list your scanner. Then see whether `scanimage -L` lists it, running it with and without *sudo*. If it only works with *sudo* then this is a permissions problem and you probably need to add yourself to the scanner group.

Q What's the easiest way to access a desktop on another Linux computer? VNC and RDP seem quite tricky to set up.

A Try *X2Go* (**www.x2go.org**), which uses SSH as its transport mechanism. This means that as long as you have SSH working, you can use it. You need the server package on the remote machine and the client package on the local system, then just run *x2goclient*.

Q How can I record the output from a terminal command so that I can look for error messages as previously suggested in *Linux Format*?

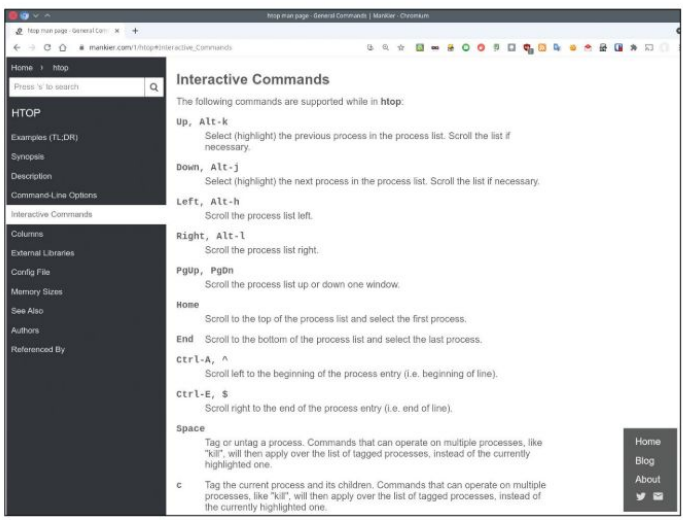
A You can redirect the output using the > symbol. `somecommand >out.txt 2>err.txt` sends normal output to one file and the errors to another. Alternatively, you can run *script* in the terminal. Nothing appears to happen, but everything you do is now recorded to a file called **typescript**. Press Ctrl+D to stop recording.

Q Why do you suggest terminal commands in so many of your answers when there are clearer graphical alternatives?

A Because it's alternatives in the plural. Each desktop, each distro, even each version, has its own graphical administration commands. On the other hand, the one on the keyboard – the terminal commands – are common across most Linux distros. They work everywhere, even if the GUI is broken.

Q It's all very well suggesting the use of the terminal, but some files have really long names and I'm a slow typist.

A The shell is able to do most of the work for you. Type the first few characters of a command or filename and press the Tab key. The shell will then try to complete the name for you. If there's more than one possibility, it'll complete as much as it can, then press Tab again. This works for directory names too, so even if a file is buried several layers down, you only need a few keystrokes. LXF



If reading man pages in a terminal is too retro for you, web sites such as mankier.com and linux.die.net provide a wealth of documentation in a more digestible format.

GET HELP NOW!

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

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

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

Mailserver

WRITE TO US

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**.

LXF014 – What about Atari?

I've noticed in your magazine that many Amiga-owning readers are concerned about the lack of support for their systems when it comes to Linux, and fair enough, the support for Amiga's when it comes down to Linux is very poor compared to PCs.

How often does this apply to Atari users then? My Atari runs Linux Debian (Potato) better and faster, than Mandrake 7.2 runs on the AMD – by quite a long way actually, but is there any support for me, and the rest of us? It took me days to actually get it installed, but when it finally did go, I was able to set everything up, even the graphics card – which it clearly states is not supported! (Yes, Atari's do have graphics cards – and soundcards, and all new ones even ISA & PCI Slots by the way!) It even lets me access both SCSI ports and the IDE line!

Damion Jones

Nick Veitch said...

The majority of Linux users run x86 architecture, so it's no surprise that that's where all the attention is focused from the distro side of things, with notable exceptions like Debian. As the Amiga and Atari platforms saw the first ports of Linux, it's no surprise that these platforms are actually well supported by the kernel.

The only time we deal with specific installation instructions is when we have a big distro on the CD (which tend to be x86 only). If we did install guides for every architecture that supports Linux, it would fill the magazine and then some. But what specific support do you want? It may be possible to do install guides on CD, but I'm not sure they could ever be more complete than the Debian documentation that already exists.

LXF028 – Hoyt mail

Hoyt Duff's column in April issue (LXF26) says: "Mail clients that can't handle HTML are simply broken". I can't agree with this. I use *Agent*, which doesn't render HTML mail, and I wouldn't have it any other way. HTML mail is conveniently shown as a blank mail, with an HTML attachment, making it easy to recognise and delete. Most HTML mail which I receive is spam and I don't like the idea of the renderer following the IMG tags (which often have identifying strings attached to the URL via a ?, thereby telling the spammer that I have read the mail). If I receive HTML mail from a known or wanted source, I reply asking them to use plain text as well as or instead of the HTML. If they're incapable or unwilling to do that, well there's always the phone.

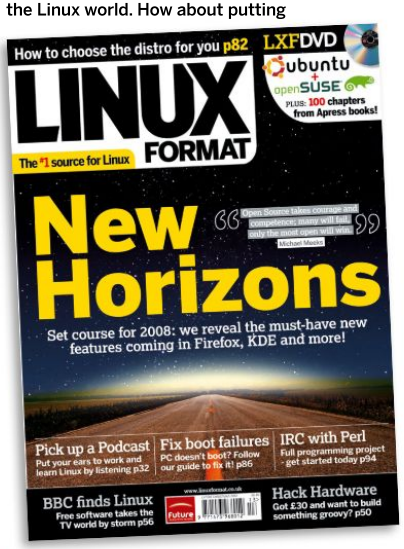
Jim Hatfield

Nick Veitch said...

The majority of the HTML email I get is spam also. Actually, come to think of it, the majority of all the mail that I get is spam. Your point about the mail agent following links which can indicate you have read the message is a good one, and I guess we should ensure that where an HTML client is used, you can stop it accessing such URLs.

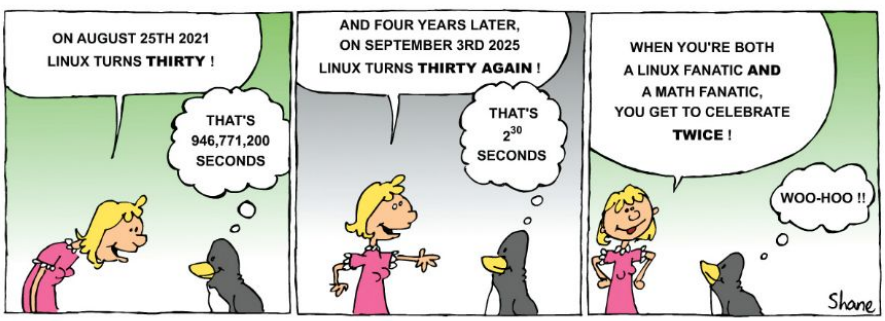
LXF050 – Linux games

Even the keenest of Linux users must enjoy playing the odd game or two and personally, that's about the only reason I have a dual-boot PC (Windows XP/SUSE 8.2). As I'm sure you're aware, commercial quality games software is a bit thin on the ground in the Linux world. How about putting



Issue 100 of Linux Format didn't feature Tux on the cover. How odd.

Helpdex



some of the Linux-compatible game demos on the DVD? The new *Wolfenstein* game for instance, would give many people a boost and is practically impossible to download on a dial-up connection (it's about 250MB).

Another suggestion for an article would be on getting ADSL up and running on Linux. I've always used dial-up at home and cannot justify £28 per month for true broadband, but nowadays some ISPs (Tiscali for one) are offering things like 150K connections for £16 per month.

Mick Scully

Nick Veitch said...

On the games front, the traditional story has been that there isn't a big-enough market for the game producers to bother releasing Linux code. Ironically, as most of them use Linux as a development environment, it isn't often really a question of the porting effort required, but the expected cost of actual distribution, support, testing and so on that puts them off.

Sadly, short of petitioning the games companies themselves, there seems to be little that can be done, apart from buying Linux games. Only when companies see a market opportunity will they make the effort...

LXF100 – Dell-ends

I thought I'd take a look at the Dell UK website to price up a new laptop. The base laptop offered is at a low price but not a very high specification, so I thought I would try a few upgrade options – I was looking forward to saving some money by not having to pay Mr Gates and Mr Ballmer. When I compared the same laptop configuration but with Windows Vista I was in for a shock.

The Vista laptop is cheaper than the Ubuntu one. With Vista, the laptop was £399, whereas with Ubuntu it was £417! What is the point offering open source software if the product is not offered at a competitive price? Linux users are used to sourcing and installing their own OS, so why pay more to not have a Microsoft licence? After all, it may be useful when it comes to passing on the PC to a novice PC user in the future...

Martin Cox

Paul Hudson said...

We're well aware of this annoyance, and have complained bitterly ourselves. The best solution is to vote with your money: if Dell wants to win Linux customers (*Dell's lovely now!–Ed*), then it needs to offer reasonable prices. Send in an email to Dell, and don't spend your money until you're happy that the prices are what they ought to be. Have any other readers had a similar experience? Let us know! LXF

» LETTER OF THE MONTH

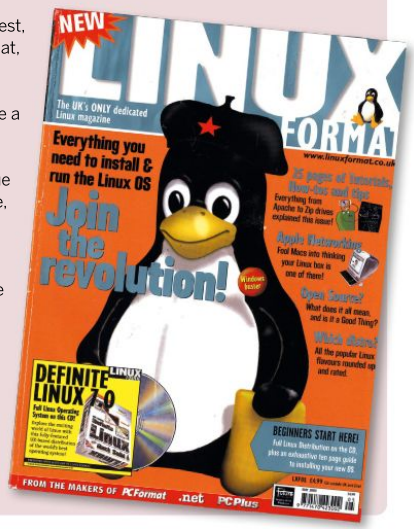
LXF001 – The road ahead

So Mr Veitch, Linux is the way forward, is it? The future alternative computing and all that? My experience of Unix-C machines only coverts the terminals at uni and I wasn't impressed, although they probably had outdated kernels and were badly configured. I never even got around to installing Linux/APUS on my trusty PPC-powered Amiga. So is it worth installing Linux/APUS and the basic Red Hat distro, which appeared on an AFCD and joining the Linux revolution? I still remain to be convinced.

Colin Buchanan

Nick Veitch said...

Linux is gaining momentum as an "alternative" OS and it's probably being installed on more computers every day than any other OS. If you know a bit about computers, Linux isn't that difficult. In many cases, the installation is what most people find trickiest, so if you can master that, you'll find the rest a doddle. A PPCAmiga running Linux would be a pretty good setup. I haven't installed Linux on my home Amiga due to a lack of drive space, but a few years ago I ran FreeBSD on a PPC A4000 and it worked very well. There's more going on in the Linux community than there has been in the Amiga world for some time now. Why not give it a try?



Comrade Tux looks rather fetching in a beret.

shane_collinge@yahoo.com

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

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

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

1) Only available to www.magazinesdirect.com subscribers

NOT FROM THE UK? Turn to page 67 for more great subscriber deals!

» CHOOSE YOUR PACKAGE!

ANNUAL PRINT EDITION

SAVE! 22%

Only **£65.00**

13 issues of *Linux Format* in print over 12 months by Direct Debit

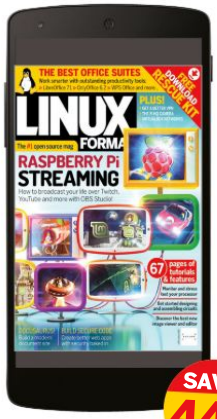
ANNUAL PRINT + DIGITAL EDITION

SAVE! 60%

Only **£70.00**

13 issues of *Linux Format* over 12 months in print and digital by Direct Debit

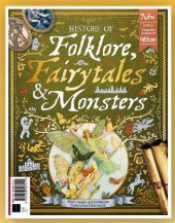
DIGITAL EDITION ONLY

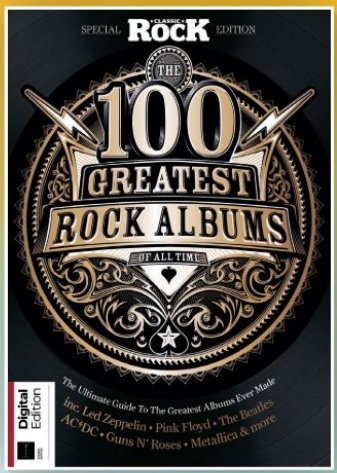


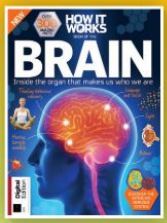
SAVE! 44%

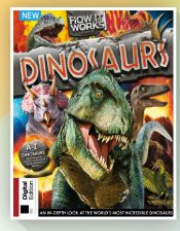
(no gift) Only **£50.00**

13 issues of *Linux Format* in digital by Direct Debit



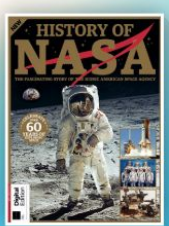






DISCOVER OUR GREAT BOOKAZINES

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.

Linux Kernel 5.14

Mayank Sharma celebrates 30 years of Linux by examining its beating heart.

IN BRIEF

The Linux kernel is a free and open source modular, multi-tasking, monolithic Unix-like operating system kernel. It was created in 1991 for the i386 PC by Linus Torvalds who still serves as the principal developer of the mainline kernel branch. Linux serves as the kernel for hundreds of open source distributions, which have come to be colloquially referred to as Linux as well.

Linux kernel v5.14 is under development, but should be out by the time this issue lands on your desk, marking three decades of what has become the leading example for collaborative software development.

From 10,000-odd lines in 1991 written by Linus Torvalds, the kernel now spans tens of millions of lines of code contributed by thousands of developers. In fact, the recent v5.13 bundled the work of 2,062 developers, making it the first release that saw the participation of over 2,000 developers in a single kernel release.

The 5.11 release of the Linux kernel had over 30 million lines of code, of which about 60 per cent were drivers. For comparison Linux 1.0.0, released in March 1994, had 176,250 lines of code.

In a sense new code is added to the kernel on a rolling basis. Each kernel subsystem is managed by a maintainer who reviews all relevant patches and queues them for submission to Torvalds within a merge window that opens for usually a couple of weeks.

Torvalds then merges all submitted patches into the source code of the prior stable Linux kernel release, creating the release candidate (marked with -rc) for the next stable kernel. Once the merge window is closed, only fixes to the newly added code in the development release are accepted. Torvalds usually goes through seven release candidates before he releases a new Linux kernel, opening the merge window for the next kernel.

The Hypocrite commit row is the most recent of the conflicts that the kernel has seen in the 30 years of its existence.

All encompassing

One of the highlights of Linux 5.14 is support for the Raspberry Pi 400. It's hard to believe that one of the most widely ported operating system kernels, which runs on everything from system-on-chips to mainframes, wasn't actually designed to be portable.

The kernel's first port was performed on the Motorola 68000 platform. The modifications to the kernel were so extreme that it led Torvalds to refer to the Motorola version as a fork. However, the move impressed Torvalds enough to restructure the kernel code to aid its porting to more computing architectures.

The kernel's hardware support has come a long way since 1991 when it was conceived and created for the i386-based PC. It now runs on over 30 major hardware architectures, with developers dropping support for out-of-vogue hardware. In fact, the 3.7 kernel series was the last one that supported the original processor.

Keeping up with the times, v5.13 became the first kernel to officially support Apple's M1-powered devices. This was the first kernel that could boot on M1-based devices, though it's still some way off from being usable for desktop users.

Perhaps the one recent change that reflects the true nature of the kernel's dexterity is the move to add the Rust programming language as a second language to the kernel. Most of the Linux kernel code is written in the C programming language. However, some kernel developers have been pushing to add support for Rust, which offers most of the flexibility and performance of C, but with added security, as a secondary language for the kernel, especially in areas where security and memory safety are of utmost importance.

The initiative has already contributed over a dozen patches totalling over 33,000 lines, which help lay the groundwork with important components such as a beta Rust compiler, an example driver and more.

While the patches were submitted during the 5.14 kernel's merge window, they weren't labelled as a pull request and will presumably not land until a later cycle.

Torvalds has referred to the kernel as evolution. Change is the one thing that's been consistent through the kernel's first 30 years and there's no indication that it's going to be any different in the future. LXF

VERDICT

DEVELOPER: Kernel developers
WEB: www.kernel.org
LICENCE: GNU GPL v2

FEATURES	9/10	EASE OF USE	6/10
PERFORMANCE	9/10	DOCUMENTATION	8/10

Rating the kernel on parameters designed to rate the software it powers doesn't make sense, just like this score.

» **Rating 8/10**

Slackware 15

Mayank Sharma has a childhood flashback to seeing his dad in the same predicament as he finds himself after all these decades.

IN BRIEF

Slackware is one of the oldest Linux distros that's still actively developed. But this isn't it's only claim to fame. The distro has some unique characteristics, thanks to being an ardent follower of the KISS principle, which endears it to its fans, even as it scares away just about everyone else. The distro officially supports the 32-bit, 64-bit and ARM architectures.

The beta release of Slackware 15 came out in April, and there's a good chance that the final release will make landfall in time to celebrate the 30th anniversary of Linux.

There isn't much of an age difference between Slackware and Linux. True, Slackware wasn't the first Linux distro; however, it didn't arrive much later and has the honour of being the oldest Linux distro that's still actively developed.

Slackware Linux started as a personal project of the then-university student Patrick Volkerding, as a more functional out-of-the-box instance of SLS Linux, and only ever saw the day of the light due to the constant pestering of his peers at the university to release his changes as a distro in itself.

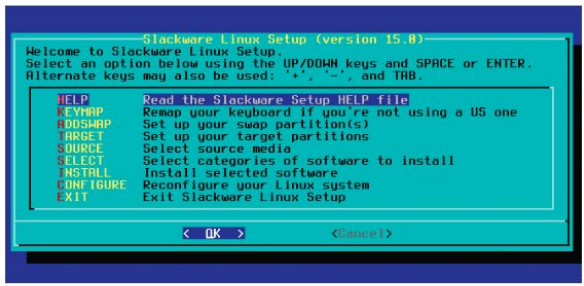
Slackware 1.0 was released on 17 July, 1993. It was built on top of Linux 0.99pl11-alpha and shipped as a set of 24 floppy disks. The distro would go on to become a working base for many Linux distros back in the day, such as the initial SUSE Linux.

The distro is popular for its glacial pace. The last major release of the distro, Slackware 14.0, was released in 2012 and even its last stable release, 14.2, is over half a decade old. Slackware's release-when-it's-ready philosophy seems out of place among the fixed and short release cycles of many of its peers.

Yet the lethargic release pace isn't a reflection of the distro's pace of behind-the-scenes development. Volkerding and a small team of contributors maintain the tree in a rolling release called "-current" and publish a release when it meets their stability goals.

The changelog for the current development version, which will become 15.0, shows a flurry of activity. For instance, while the 15.0-beta image shipped with a 5.10-series kernel, this has already been upgraded to a 5.13-series kernel in the -current branch.

The other notable presence is that of the Wayland display server. While the display server has been added to -current, the venerable X hasn't been dropped and given the distro's preference for stable, well-tested software, X could still be the default display server in Slackware 15. The one thing you can be sure wouldn't be part of Slackware 15 is the *systemd* service manager, with the distro seeing no reason to replace the SysV *init* system, which has served it well through the years.



Slackware exists in a totally different universe where publicly developed projects don't have a public bug tracker, a code repository or even a well-defined method of community contribution.

Kiss the frog

Everything about Slackware feels drab and dull, especially compared to its peers that are always keen to toot their shiny new bells and whistles. For instance, while modern distributions are tripping over each other trying to sing praises about their installers, Slackware still proudly carries an ncurses-based one from the last century.

Another characteristic of the distro is its lack of graphical administration wizards and tools to handhold users through administration and setup tasks. Oh, and its package management system doesn't resolve dependencies, either.

And all this is a deliberate design decision, and perhaps the main reason behind Slackware's popularity. Slackware takes pride in helping users exercise complete control over their installation. The distro's philosophy is that interacting with it will give users an insight into Linux that you can't get anywhere else. In the same vein, the distro's approach to package management – which essentially involves compiling them from source – is a reflection of its ethos of technical simplicity.

All things considered, while there are still quite a lot of fans of Slackware's philosophy of simplicity, the fact is that setting up and maintaining the distro requires a lot of time and effort, which isn't a luxury everyone can afford, either on the desktop or on the server. LXF

VERDICT

DEVELOPER: Patrick Volkerding
WEB: www.slackware.com
LICENCE: GNU GPL

FEATURES	**7/10**	**EASE OF USE**	**5/10**
PERFORMANCE	**9/10**	**DOCUMENTATION**	**6/10**

Slackware is incompatible with our rating system. Anyone who uses Slackware wouldn't use anything else.

» Rating 6/10

Debian GNU/Linux 11

Mayank Sharma can't help but marvel at one of the oldest Linux distros, which continues to be as riveting as it was almost three decades ago.

IN BRIEF

Debian GNU/Linux is one of the oldest Linux distros that's still actively developed. It was designed by Ian Murdock in August 1993 and had its first release in September that year. Upon release Debian was sponsored for a year by the Free Software Foundation, before forming its own non-profit organisation named Software in the Public Interest (SPI), which has since sponsored dozens of influential open source projects.

Debian 11 (codename Bullseye) has just entered full freeze, and while the final release might not have arrived even by the time you're flipping through this issue, it's surely just around the corner.

Unlike some of the other distros that bring in a raft of new changes, Debian releases are conservative by nature. So while Bullseye has a ton of new things compared to Buster (which came out in 2019), most if not all have already made landfall in several other mainstream distros.

Then there's Debian's release and support cycles, which is a far cry from the one adopted by most desktop distros. While the likes of Fedora and Ubuntu will put out multiple releases in a year, and support them for between nine and 13 months, Debian supports its releases for five years. Furthermore, Debian 12 won't arrive for at least two years, which means the distro will look increasingly outdated.

One of the best things about Debian is its extensive software repository and Debian 11 is no different. Since Debian 1.1, which included 486 packages, the total number of packages in each Debian stable version has grown exponentially. In fact, Bullseye includes over 13,370 new packages compared to Debian 10, bringing the total number of packages to over 57,000.

Behind the scenes, Bullseye introduces driverless printing and scanning thanks to the support for the vendor neutral IPP-over-USB protocol, which extends the advantages of driverless network printing to devices connected via USB.

The release also ships with the Linux 5.10 LTS kernel, which will be supported until 2026. One of the advantages this kernel brings to Debian is built-in support for the exFAT filesystem.

On the desktop front, Debian 11 as usual ships with several desktop environments, and you'll find virtually all open source graphical applications in its repos. It's also worth noting that Debian 11 supports a total of nine architectures, including the 32-bit i386 that's been abandoned by most other distros.

Principled approach

In addition to taking on new packages, Debian 11 has also dropped over 7,000 packages from its repositories for various reasons. Notable exclusions include the *LILO* boot loader, which hasn't kept up with the times and has been replaced by the more feature-rich *GRUB* boot loader. Bullseye also doesn't include the *Chef* configuration manager, because of copyright issues.

Perhaps the one aspect that's helped define Debian, but has also led to some notable disagreements in the project, are its governing principles as outlined in its constitution, social contract and policy documents.

From the very beginning the project has adopted a disciplined approach as mentioned in its social contract,

Debian defaults to the Gnome desktop, though Debian 7 switched to Xfce during the early days of Gnome 3, before reverting to the mainstream desktop with Debian 8.

which has created friction both outside and inside the project. For instance, Debian was involved in a decade-long cold-war with Mozilla, that started when the distribution refused to use the *Firefox* logo because it was incompatible with Debian's free software guidelines. It led to the creation of the *IceWeasel* browser, together with several other Ice-prefixed forks of Mozilla software.

The other major rocking-the-boat moment for the project was the switch to Systemd system and service manager, which led to the birth of the Devuan distro.

The use of Systemd resulted in a long, technical and much-publicised debate that caused many notable Debian developers to tune down their involvement with the project, including Ian Jackson, the author of Debian's *dpkg* package management tool.

Still, even almost three decades of development have been unable to rob Debian of its enigmatic charm and the distro continues to be managed with the same passion and fervour as it was when it came into being in the previous century. LXF

VERDICT

DEVELOPER: The Debian Project
WEB: www.debian.org
LICENCE: DFSG-compatible licenses

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

A project that's more than the sum of its developers, Debian – despite its age – shows no signs of slowing down.

» **Rating 8/10**

Fedora 35 Rawhide

Mayank Sharma is pleasantly surprised to discover that this distro, which can trace its roots back to the early days of Linux, hides another one in plain sight.

IN BRIEF

Fedora Linux has become one of the most popular Linux projects that produces distros to cater to all the popular Linux use cases. The distro can trace back its lineage to Red Hat Linux, which came out in the mid-1990s, and helped the now IBM-owned Red Hat become the first billion dollar open source company.

The latest version of Fedora Linux 34 was released at the end of April 2021 (check our review in **LXF277**) . However, it branched from the testing branch dubbed Rawhide, back in February, which marked the beginning of the development stage for Fedora 35.

Although Fedora 35 won't arrive before late October or early November, the Fedora Engineering and Steering Committee (FESCo) has already approved a large number of proposed features for the release, so we have a good idea of what the upcoming distro will bring.

If you have the chops to troubleshoot issues, Rawhide is a wonderful platform for you to test bleeding edge features long before they end up in a stable Fedora release.

One of the changes is the move to *WirePlumber*. While Fedora 34 shipped with *PipeWire* for managing audio/video streams, replacing *PulseAudio*, with Fedora 35 the distro will use *WirePlumber*, which is touted as an advanced *PipeWire* session manager. Don't worry too much if this is the first time you've heard of *WirePlumber*. Fedora's raison d'être is to innovate by introducing cool, new stuff. It came into being as a rapidly moving platform, and by its nature moves quickly, with a new release every six months.

Fedora Linux started in 2002 as a volunteer project to provide well-tested third-party software packages for the Red Hat Linux (RHL) distro. RHL had its first release in May 1995 and rose to become one of the first commercially successful Linux distro. When Red Hat realised it can't stabilise and innovate on the same distro, it discontinued RHL and created Red Hat Enterprise Linux (RHEL) as its paid enterprise-grade distro, while it merged RHL with Fedora Linux repository to launch the officially supported, community distro, known as Fedora Core Linux.

Strength in unity

Fedora Core had two main software repositories: core and extras. One was maintained by Red Hat and the other by the community. The distinction was dropped starting with Fedora 7, when the two repos were merged and put under the guidance of the community.

The next major step in the evolution of the distro came with the release of Fedora 21 in 2014. With the release, the single distro was split into three editions, namely Workstation, focused on desktop use cases; Server for server deployment; and Atomic, designed for cloud computing.

Fast forward to 2019's Fedora 30, which persisted with the Workstation, and Server editions, but replaced Atomic with CoreOS to focus on cloud computing deployments. The release also added a couple of more editions to the Fedora lineup, in the form of Silverblue, which focused on an immutable desktop specially designed for container-based use cases and IoT, designed for IoT devices.

Development of Fedora takes place inside Rawhide, which is both the name of a repository and a distro. Package managers integrate the newest usable versions of their packages inside Rawhide, and at the end of every day a snapshot if the repository is compiled into an installable distro. In addition to being a useful platform for package maintainers, Rawhide also serves as an excellent rolling release for the advanced users willing to keep up with its continuous flow of updates.

Red Hat saw Fedora as a means to give new features a platform to stabilise before they could be introduced in its long-supported enterprise-grade RHEL. So for instance, Fedora 12 became the base for RHEL 6, Fedora 19 for RHEL 7, and Fedora 28 for RHEL 8.

The introduction of CentOS Stream has added another stepping stone between Fedora and RHEL, but Fedora continues its tradition of giving users a bleeding-edge showcase of the best Linux innovations. LXF

VERDICT

DEVELOPER: The Fedora Project
WEB: www.fedoraproject.org
LICENCE: Various

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

Fedora is an easy-to-use option for those who want to experience the bleeding edge of Linux for a range of uses.

» Rating 8/10

OpenSUSE

Mayank Sharma is glad that being able to correctly capitalise its name isn't a requirement for running the distro that's almost as old as Linux itself.

IN BRIEF

One of the leading RPM-based distros, openSUSE is a direct descendant of the distro that sprang into life not long after the kernel. One of the main reasons for its survival has been its willingness to mould itself according to the needs of the ecosystem it serves. It's also ensured its longevity by making itself invaluable to its corporate sponsors, without compromising the interest of its community of users.

openSUSE only came to life in 2005. However, its legacy dates back to the early days of Linux with SUSE, which was a company before it was a distro. SUSE was founded in September 1992 in Germany, and SUSE 1.0 came out in 1994, which was a localised German version of Slackware. It took another couple of years before SUSE released its real distro in 1996 after hiring the lead developer of the defunct Jurix distro.

SUSE was always about monetising Linux, be it in terms of support or boxed distros. In 2003 SUSE was acquired by Novell, which rolled out the openSUSE project with the intention of involving the community with the development of the distro. Novell was subsequently acquired by Attachmate, and changed hands several more times, though all the corporate juggling has had little impact on openSUSE.

The distro continued its rise in popularity and in 2014 the developers split off a stabilised instance of its development branch named Factory into a rolling release distro: openSUSE Tumbleweed. This forms the base for SUSE Linux Enterprise Server (SLES) and Enterprise Desktop (SLED). With one aspect of the Linux use case sorted, the developers focused on delivering long-term stability with the introduction of openSUSE Leap in 2015, which borrows core elements from SLE. Aligned with the sensibilities of LTS releases, Leap puts out one minor release once a year that's aligned with SLE Service Packs, and one major release between 36 and 48 months, aligned with the major SLE releases.

If you look at the bigger picture, you'll realise that unlike Red Hat, which uses both Fedora and CentOS Stream to feed into Red Hat Enterprise Linux, SUSE has positioned its two openSUSE releases on either side of their enterprise offering. The seeds of separation between the two versions were sowed back in late 2010 by veteran Linux kernel developer, Greg Kroah-Hartman, who at that time was working for Novell. Originally, he proposed Tumbleweed not as a distro in itself, but rather a repo of rolling updates that could be installed on top of the regular openSUSE release.

openSUSE's installer offers an interesting option called System Roles, which are predefined use-cases to help deploy the distribution for a particular purpose.

Open Sesame

One of openSUSE's marquee components is the *YaST* setup and configuration tool. *YaST* (*Yet another Setup Tool*) is one of the tools from SUSE of the 90s that's been refined over the years to handle everything from installing the distro to administering an openSUSE installation. In fact, thanks to *YaST*, openSUSE Leap 15.1 became the first multi-purpose operating system that supported the same Linux installation experience from the desktop to the Raspberry Pi.

The 15.1 release also introduced the Transactional Server, which uses a read-only root filesystem and comes from the Kubernetes on SUSE project, Kubic. This feature includes an update system that applies updates as a single operation, which makes it easier for users to do Btrfs rollbacks for multiple packages.

YaST's package management aspect is managed by another openSUSE speciality, the *ZYpp* package manager. It came about after the SUSE's acquisition by Novell, which had just taken over Ximian. The two open source companies brought different package managers (*Red Carpet* and *YaST*) and *ZYpp* came about as Novell's merger of the two.

SUSE has moved from a boxed distro, and a closed development model to a free distro and an open development model. Corporate upheavals notwithstanding, openSUSE has forged a symbiotic relationship with its sponsors' enterprise offering that ensures the continuous existence of both. LXF

VERDICT

DEVELOPER: OpenSUSE Project
WEB: www.opensuse.org
LICENCE: Various

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

The openSUSE stable of distros is a comprehensive one that offers something for all users for all popular use cases.

» **Rating 8/10**

Mageia 9

Mayank Sharma remembers Linux-Mandrake 5.1 for sowing the seeds of his complex relationship with KDE. How relevant is its successor in 2021?

IN BRIEF

The Mageia Linux distro is one of the most popular forks of Linux-Mandrake, which is seen as the distro that helped make Linux take big strides towards the desktop. Linux was poised to become the next big thing in the server space, but it struggled to make any headway on the desktop. Linux-Mandrake polished Linux's usability and despite its collapse, its legacy lives on in Mageia.

Mageia is one of the newer Linux distros, but it has roots that extend back well into the '90s. Innovating the Linux desktop has been the driving force behind the distro since its very beginning.

Many consider Mageia as the spiritual successor of the distro that came into existence as Linux-Mandrake. The brainchild of Gaël Duval, Linux-Mandrake combined two of his favourite things in Linux: the RPM packaging format and the KDE desktop environment.

From the get-go, Linux-Mandrake's mission was to enhance the desktop experience of Linux, combining behind-the-scenes performance tweaks with desktop conveniences, such as the ability to auto-mount CDs without messing around with configuration files.

It didn't take long for users to throng to Linux-Mandrake, which led Duval and some other developers to create MandrakeSoft, with the intention of distributing Linux-Mandrake media. Buoyed by their initial success, MandrakeSoft went public in 2001. However, the company couldn't keep the cash registers ringing and in January 2003 filed for the French equivalent of bankruptcy protection.

Just about then, Red Hat decided to discontinue their desktop-oriented Red Hat Linux product, and Linux-Mandrake welcomed its users with open arms. In fact, at the end of 2003, MandrakeSoft announced its first quarterly profit and, in March 2004, a French court approved its plan to emerge from bankruptcy and return to normal operations.

Over the years, MandrakeSoft acquired a host of companies, including the Brazilian Linux distribution Conectiva, after which it changed the name of the company to Mandriva and its distro's name to Mandriva Linux. In 2006, after several more acquisitions, including the Lycoris distro and the enterprise software infrastructure company Linbox, Mandriva laid off several employees including Duval, the company's co-founder and lead developer.

Win some, lose some

The company, however, struggled to keep its balance sheet in the black. Mandriva sold a controlling stake to a Russian company called NGI, and announced that while it intended to continue working on its server products in Europe, development of its desktop distro would be moved to one of the BRICS (Brazil, Russia, India, China, and South Africa) countries. Following the deal, Mandriva liquidated its subsidiary Edge-IT and laid off its employees, who then went on to fork the distro into the community-oriented Mageia.

Mageia's founders have continued the Linux-Mandrake tradition, though they've been clear that theirs is a community-controlled, community-governed project. Mageia still rocks the custom tools Linux-Mandrake first wowed desktop users with, such as *DrakConf*, the all encompassing Control Center that inspired a generation of one-stop configuration panels.

The one major difference between Linux-Mandrake and Mageia is that while the former focused on KDE, the latter claims to be free of any such bias for a particular desktop.

Apart from the behind-the-scenes economic upheaval at Mandrakesoft, other factors also played a role in the decline in popularity of Linux-Mandrake, by then known as Mandriva. Chief among them was Ubuntu's release in 2004 as a desktop distro that promised as much convenience as Mandriva, with an easy-to-use installer and simplified system administration, but without the costs, was a turning point for desktop Linux as a whole, and Mandriva in particular. People were no longer lining up to buy boxed sets of Mandriva Linux, when Canonical would ship one to you for free anywhere in the world.

Yet despite the collapse of Mandrakesoft's business model, the contribution of Linux-Mandrake to accelerate the adoption of Linux on the desktop can't be ignored. The distro leaves behind a rich legacy and a handful of descendants, of which Mageia seems best aligned with the sensibilities of the modern open source desktop ecosystem. LXF

VERDICT

DEVELOPER: Mageia Community
WEB: http://mageia.org
LICENCE: GPL and others

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

Mageia continues to be one of the best desktop distros, which continues to innovate while staying true to its legacy.

» Rating 8/10

Arch Linux

Mayank Sharma finds himself at that juncture in his life where he has the time to appreciate Arch, but not the patience to use it as his daily driver.

IN BRIEF

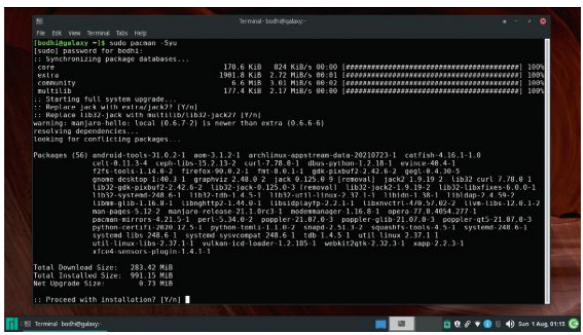
A rolling-release distribution that officially supports only the 64-bit processors, Arch Linux is often cited by power users as their favourite Linux distro. Its strong focus on simplicity in design, which leads to versatility in operation, makes Arch a DIYers paradise. The distro can appear cumbersome to desktop newbies, while being straightforward for experienced Linux users.

From a chronological perspective, Arch Linux is a relatively new distribution. But in the grander scheme of things, it's one of the few distros that despite not being born in the same century as the Linux kernel, has had a profound impact on the Linux distro ecosystem.

The design approach of the distros' development team focuses on elegance, code correctness, minimalism and simplicity, which calls on the user to be willing to put in some effort to understand its innards.

Judd Vinet started the Arch Linux project in March 2002. He was inspired by the source-based CRUX distro, but didn't take a fancy for its package management. This led him to create the package manager *pacman*, which soon became one of Arch's crown jewels.

Back in 2002, most distros were compiling against the i386 architecture, while Arch was optimised for the i686. This staying-ahead-of-the-curve became a hallmark of the distro, which selected an install-once-update-forever, rolling-release model. Arch added PAM support in June 2003 when it was still very new and experimental, and replaced XFree86 with XOrg soon after its release in 2004.



Earlier in 2021 Arch started bundling a new guided installer to help simplify the notoriously cumbersome Arch installation process.

Best of both worlds

Over the years Arch has come to be the favoured distro for people who want to be as close to the bleeding edge as possible, without the complexities of compiling everything from source.

Arch is the distribution of choice of long-time Linux kernel developer Greg Kroah-Hartman, and is usually the first one to make the latest Linux kernel release to its users. For instance, Linux 5.13.6 and the 5.10.54 LTS version came out this year on July 28 and were pushed to Arch users on July 29. The trend continues with virtually every other piece of software as well. *LibreOffice v7.1.5* was available the same day it was released, and *Firefox v90.0.2* was available within a few hours as well.

Arch Linux focuses on simplicity in design, which it adheres to by shipping software as released by the original upstream developers with minimal distro-specific downstream changes. It also takes special care to use well-commented, easy to follow configuration files, which endears it to proficient Linux users who have the skills to appreciate its DIY nature.

The *pacman* package manager is one of the major distinguishing features of Arch Linux. It combines a simple binary package format with an easy-to-use build system. *Pacman 2.0* in July 2002 debuted the initial script for the Arch Build System (ABS), which is the magic ingredient that makes it easy to turn source code into a compiled package.

A couple of months later, *Pacman 2.1 2002* introduced support for multiple repositories. The current repos took shape in 2007, which is also the year when the developers decided that all packages that end up in the Core repository must go through the Test repository to ensure that they don't end up bricking an installation.

In April 2005, Arch introduced another of the distro's highlights, the Arch User Repository (AUR), which provides user-made **PKGBUILD** scripts for packages that aren't included in the main Arch repos.

In addition to the distribution itself, another of Arch's hallmark contributions to the broader Linux landscape is the invaluable Arch wiki. Since going online in 2005, the Arch wiki is brimming with guides and fixes that can just as easily help chaperone users on virtually any Linux distro. LXF

VERDICT

DEVELOPER: Arch Linux Developers
WEB: https://archlinux.org
LICENCE: Various

FEATURES	**9/10**	**EASE OF USE**	**5/10**
PERFORMANCE	**9/10**	**DOCUMENTATION**	**9/10**

Arch doesn't give newbies point-and-click conveniences, but it's one of the most well-documented power-user distro.

» **Rating 8/10**

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

Roundup

Elinks 0.13.2 » w3m 0.5.3 » Links 2.21 » Lynx 2.9.0dev.6 » browsh 1.6.4

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

Terminal browsers

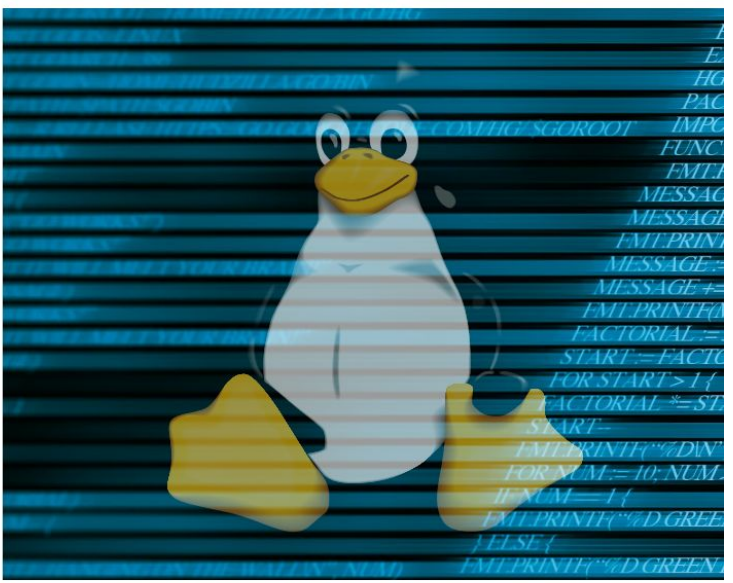
Web browsers are heavy on resources and hard to tweak. **Mats Tage Axelsson** identifies five that you can customise to your heart's content.

HOW WE TESTED...

To explore the capabilities of these terminal-based browsers, we used them to access popular websites and tried to download data where applicable. We also attempted tasks that you would normally carry out on those sites. Once we had accessed a site, we made sure that you could find articles or files easily.

With the many different ways to interact with other programs, we also tested how to download media and run it on your system. Looking beyond daily computing tasks, you may want to download data and process it, so we tested how to obtain the information in the correct format for use in a set of analysis programs.

The amount of tasks you can carry out within a terminal-based browser is impressive, especially considering that most sites use performance-intensive method to display information. There are many standards that these browsers don't support, but also a few that only these can support. Can you use Gopher in your regular browser, for example?



Today's web is full of graphics and tricks to deliver a great experience. All of this razzmatazz does cause some people to stop and wonder what it's all for, when all you want to do is ask a simple question or look up a simple fact. Fetching all this data will take many times the data to make the site look great, wasting resources and your patience.

If you're usually on the command line anyway, using a text-based browser can make these tasks much faster. The enormous challenge is that JavaScript is usually needed for many sites. Also consider that content is arranged using CSS, which takes even more CPU and memory from your system. This begs the question: is it even possible to use a text-only browser without fancy tech to get anything done on the web?

With some simple scripts and careful choice of switches, you can achieve a surprisingly amount without the bloat of a modern browser. If you put a little bit more effort in, you may be able to scrape data from sites and arrange it locally. There are also some obscure protocols such as Gopher that you may be able to use with these browsers, and even BitTorrent has a place here.

Installation considerations

Some of these programs need to be compiled from source.

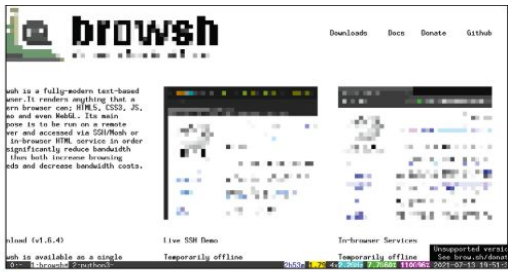
There are several reasons for wanting to use a terminal-based web browser. Often it's because you don't want to leave the command line. Other times it's because you don't have a graphical user interface, which is the case when you're working over *ssh*. In either case you need to make sure you have the correct software installed.

By default, *w3m* is usually present on Linux systems. The others in this month's *Roundup* are usually available from repositories or in some cases only as source code. Installing the other browsers is done through with your package manager. *Lynx* has very few dependencies, while *w3m* has more because it supports images. There are also more options, including *w3m-el* so you can run it inside of *Emacs*. Of course, you can use *eww* for that too, which is a module for *Emacs*.

When you look closely at *elinks*, you'll find many more dependencies. This is because the program supports many more standards. It also supports browser scripting and CSS. This also makes *elinks* one of the most capable browsers of all the programs on test this month.

To get as much of a graphical browser experience as your terminal can support, you need to look to *browsh*. This browser only depends on *Firefox*, but if you want to use it remotely then *Mosh* is the natural choice.

Even when choosing a text-based browser, you may still be interested in performance. So, can you compile on your own



Browsh has all the graphics capability of Firefox, so you need to tweak the terminal to handle the resolution. You need to re-compile your terminal to support TrueColor.

easily? If you look through the code, you'll quickly realise that these are mostly C or C++ with the exception of *browsh*.

To compile *Lynx*, use the regular `make` command. You have a **makefile** in the C++ directory. Nothing is odd about this build if you've compiled anything for your Linux box yourself. *Links*, *Elinks* and *w3m* also have C-code, a **makefile** and in the case of *w3m* some *awk* to set your environment up.

The story for *browsh* is different to say the least: you need an *npm* install, Webpack and *Firefox 56* or higher. For web developers and, frankly, anyone beyond beginner level, this shouldn't pose a problem, but it's important to bear in mind if you want to compile inside a virtual private server.

VERDICT

ELINKS	**8/10**	**LYNX**	**6/10**
W3M	**7/10**	**BROWSH**	**5/10**
LINKS	**6/10**		

Browsh receives the lowest score here, but mostly because it's different from the competitors, because it uses web-based tech.

Use and capabilities

Since you're in a terminal, are the limitations too much to stomach?

Many of these programs are old. Although people are still maintaining them, new features aren't a priority. You'll run into some problems with sites that require JavaScript, advanced colour schemes and scripting. Having said that, you can do without these features in many more cases than is apparent when you have it available at all times.

Working in *Lynx*, you have only text, no JS and you must start the program with a URL. You can't open a menu to help you navigate the features. The development of *Links* has led to a slicker interface, drop-down menus and other snappy features. You don't have to use them, but they are available. It handles frames, background downloads and graphics to a certain degree, too.

Elinks' developers went a step further with menu possibilities, tabs, cookies, CSS support and browser scripting. It's close to having a full browser on the terminal.

For the *w3m* team, functionality – including supporting other programs – is important. When browsing in *w3m* the program can

Menubar will give any terminal-based program a polished look. Surfing and changing settings when you start out is an absolute breeze.

handle tables, frames and more. *W3m* works as a pager, so you can read HTML emails using it. You can set *w3m* in your file manager as a preview image option, too.

Because it's using *Firefox*, the *browsh* terminal comes with the most bells and whistles. If you have some mastery in compiling your code and handling early software releases then it's worth looking into using *MoSH*, the mobile shell.

VERDICT

BROWSH	**9/10**	**LINKS**	**6/10**
ELINKS	**8/10**	**LYNX**	**5/10**
W3M	**8/10**		

Here browsh comes top because it's had help from Firefox. Is this cheating?

Usability straight out of the box

How painless is first-time use?

A text-based web browser will probably not be something you base your online life around. For this reason, you'll need to get started quickly without too much configuring. So what do you need a text-based browser for, anyway?

Mostly, you want to be able to use it for finding answers to pressing questions about what you're doing right now. Using a form, and reading advice and help from your peers is one way of doing that. A quick search using your favourite search engine is a necessity, so filling a form and getting the result back is crucial. To be able to stay on the command line, you must also be able to find the results.

Showing the results on the terminal can be confusing when frames and columns aren't well supported. Some of these browsers support these elements better than others. You can save a lot of scrolling if your browser gets this right.

Elinks 9/10

Elinks has a pleasing appearance and you can customise many of its features. You can open the menu bar using F10: it's a good way to become familiar with aspects of the program. Within the menu are all the main features and keybindings. You can create your own keybindings, too. You can also set it to run an external command. Watching video is an example: just choose a media player, then use the keybindings and URI Redirection functions.

Search engines and help sites all assume that you're using a graphical browser. *Elinks* will handle the different ways columns work very well, although when using Stackexchange you'll need to scroll quite far. Search engine results are handled gracefully and you can easily follow the links.

Overall, *Elinks* works like a charm considering that the web is dominated by graphics and visual tricks.

w3m 8/10

W3m features less clutter on its interface, but whether you think that's an advantage or a handicap will depend on how you work. There's no menu to access in case you have doubts about how a command can handle a link, for example.

Search engines give very clear results, Stackexchange and similar sites look good, though you'll still experience scrolling issues. Filling in forms is direct and simple; hit Enter while in the input box and then start typing.

Since you're missing the menu, you'll have to find the help file (H) and read through the list. *Vim* and *Lynx* users will be used to most keybindings, though. You can also set external browsers for the times when the command line isn't enough.

The two strongest points of *w3m* is that it works as a pager and can handle graphics really well. Because it's a pager, you can also read your emails.

Graphics capability

Graphics aren't vital but remain of interest when using a terminal browser.

Graphics capability covers several angles, such as how frames align text and if images are displayed correctly. You won't see it done perfectly in terminal browsers because since they aren't designed for it in the first place. You may also want to have hints and help on display.

Modern web sites usually use CSS to create order on the page. This means that without it, the site will need to be scrolled through when it's supposed to be organised into columns. Stackexchange is an example where you need to scroll over two screens to get to the answer you're looking for. You'll find the conversation after some effort, but it will not look as intended.

Browsh's designer has made a conscious decision to handle this in an unconventional way. It renders CSS using *Firefox* and sends an estimated rendering to your terminal. For this to work really well, the terminal must be TrueColor. This method makes it possible to watch an approximation of video in your terminal. It looks awful because it renders it using a custom font to represent the different pixels.

W3m handles graphics well, but it doesn't do anything about organising the tables, frames and columns. This makes it a little messy on many pages. Its user interface is sparse.

Elinks looks great at startup. The status bar has nothing on show, but hit F10 to access help files. *Elinks* doesn't display images, but you can define an external viewer. You can also do this for video. *Links* is the same in this regard – not surprise, as the two browsers are forks of the same code so many things are similar. You can also set up a viewer separately. *Lynx* ignores this issue. All browsers except *browsh* hands video over to other programs.

VERDICT

BROWSH	**9/10**	**LINKS**	**6/10**
ELINKS	**8/10**	**LYNX**	**3/10**
W3M	**7/10**		

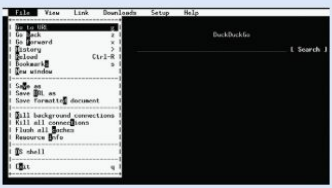
Despite being the best in this category, browsh has serious limitation. It requires TrueColour terminals, which you have to compile yourself.

Links 7/10

Links features the same drop-down menu as *elinks* – press F10 to activate it. It also handles scrolling about as well as *w3m*, which is to say, not very well at all. Keybindings make navigation slightly different, You'll use the cursor keys more and press X and Z to handle links. While this is a handy way to view websites, it's not the easiest of methods for *Vim* fans. The choice of keybindings is rather odd for these types of browsers, but with a little bit of time and effort they're easy enough to learn and understand.

Because CSS is not well supported, some pages will look odd. We have the scrolling problem all over again. In most cases the result is clearly readable, though.

When you realise that you're in over your head and need a GUI, *Links* comes to the rescue; It also runs as a GUI when you open it from the application menu of your X environment.



Lynx 6/10

Lynx is the old man on the block (*I know how that feels – Ed*). Oddly enough, this may be the reason why the browsers feels blisteringly fast. You can choose the keybindings from the Options menu. To help you find that menu and any help you may need, you have some of the current keybindings at the bottom of the screen at all times. You can choose to turn on *Vim* and *Emacs* keybindings at your leisure. Back always goes back to a list of old links.

There's no support for CSS or JavaScript, so a good number of sites will break. However, using *Lynx* when it does work is very fast. All the scrolling issues that the others have are the same here.

All images are ignored, linked to or displayed as links. You simply have no graphics support. Despite this, the experience is impressive because you can read the text you want quickly, before moving on to the next site.

browsh 5/10

The keyboard help is missing in *browsh*, though the list is short and you can find it quickly on the developer's web page. With this list it's very easy to get started. All web pages will display well – even the ones using CSS. The slightly odd part is that *browsh* doesn't parse CSS; instead, it prepares the page from what *Firefox* generates.

To follow links, you must use the mouse to point and click them. Any keybindings are there to select and manage tabs, history and getting to the URL bar. Functionality is sparse and graphics usually work, but look ugly.

The primary reason to use this terminal browser would be to get enjoy the reduced bandwidth – a by-product of the basic graphic display. To get full use of that, you must configure your terminal to be TrueColor; the most common setting is 256 colours, which isn't enough.

Modern sites, JavaScript and more

Without JavaScript, many web sites look awful or simply don't work.

Many websites require JavaScript to run at all times. Other question and answer sites, like Stackexchange, run fine without it. Another standard that has a profound effect on web pages is CSS, which is mostly used to style the page. This includes placing the content in columns.

From the browsers on test here only *elinks* supports CSS and JavaScript, although *browsh* has all the functions supported by *Firefox*. *Browsh* is an exception for many cases. In the end it renders it all for you after having had it all handled by *Firefox*.

The *Lynx* browser supports neither CSS nor JavaScript and it probably never will. You get speed and agility but not using these functions. The developers will most likely consider it an insult if you ask for this functionality to be introduced!

W3m, despite having a more modern appearance, doesn't support JavaScript or CSS. The developers list the lack of this support as a feature because the script often takes much more bandwidth than the information you want. Rendering CSS is also a very CPU-intensive activity.

Links can support JavaScript, but you have to compile it yourself. The support was designed early on in the browser's development, but it was buggy so it was turned off by default. CSS isn't supported, and it only renders HTML 4.0.

The *browsh* browser ignores the issue by letting *Firefox* take care of it. This means that you can run any site and it will be well rendered. However, because of how it's rendered, you'll invariably see big blotches on any visuals. Considering all this, you must decide what you're going to use your browser for, and whether you're happy going down the text-only path.

VERDICT

BROWSH	**10/10**	**LINKS**	**3/10**
ELINKS	**6/10**	**LYNX**	**3/10**
W3M	**3/10**		

Not supporting these standards is a deliberate choice, but sometimes the website depends on it. Choose your terminal browser carefully.

Functions and features

These browsers are designed to be agile – you add your own features.

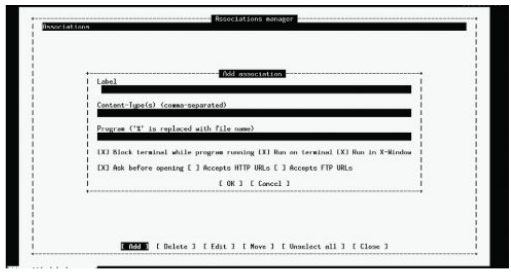
At first glance, these browsers will just show text from web pages – and badly at that. Yet you can use these browsers for many small tasks, although many sites will not work properly.

Lynx's deliberate effort to remain lean has paid off. It does support local editing, so you can change what you see. You can define execution links and set your personal Mail Address so you can use that to send email and fill in forms. It even has a feature that shows transfer rate. If you're still using Usenet, you're in luck. The Gopher protocol is making a comeback in some circles, and *Lynx* has got you covered.

Links and *Elinks* are related, so they work in similar ways. They both support a mouse (using *gpm*) and *Mime*-type calls function, where you can call other programs according to the MIME type. *Links* is the only browser that has a cache for formatted pages.

Elinks also supports tabbed browsing. The menu system is the same as *Links*. The support tables and frames aids rendering, and you can use BitTorrent if you can compile the protocol yourself.

W3m works well with simple web pages but it's also a surprisingly good pager and directory browser. Use it on local files and email and you have a solution for many tasks. This includes image viewing, but for more advanced jobs you'll need to hand over to other programs. Video is a great example. You can also compile *w3m* with mouse support.



Configuring how Elinks handles special MIME Types. You can pick any program to make this happen – even your own script.

Finally, *browsh* supports tabbed browsing, and thanks to its connection with *Firefox* also supports all modern JavaScript and CSS by not doing it on its own.

VERDICT

ELINKS	**8/10**	**LYNX**	**5/10**
W3M	**7/10**	**BROWSH**	**3/10**
LINKS	**6/10**		

Winning in this category is not necessarily better from the perspective of most terminal users. Media is of lower importance.

Working with other programs

When you choose one of these browsers, you end up wondering how to handle edge cases.

When you choose a terminal-based web browser, you probably have an ulterior motive. Speedy downloads is great, but when you need to access data quickly for processing, tools that can download quickly is essential. In most cases, you can use programs such as *cURL* or *wget*. However, they usually lack the interactive capabilities.

Some of these browsers can send login data when run as a script. Sometimes, you just need one graphical tool for a very specific need.

Lynx can't do much with other programs, but it's an excellent tool for fetching data from web pages. You can call it with many switches, controlling what comes out and adding user names. This level of control can help you filter the data before you receive it. The dump command outputs pure text, not HTML.

While running *Lynx*, local execution of programs is normally off the table. However, you can use `lynxexec` and `lynxprog` to launch other software.

You can also define an external editor for forms. *Links* uses the same command to dump formatted content. You can call other programs using the MIME-type and file extension, a capability shared by *Elinks*.

The wide array of options for Elinks makes for a powerful tool for more advanced web usage. It's also powerful for scripting, with many options for output.

w3m enables you to choose the format of the output, giving you a file in formatted, HTML or HEAD data. You can also combine the two. When you run *w3m*, you can also call external programs using MIME and extension. It can also post when called by a script, enabling blogging using the command line. Since it works as a pager, you can even use it for comparing files. *browsh* shows web pages to the best of its ability, by interacting with *Firefox*.

In this month's *Roundup*, you can see that your choice will depend on what you want your text-based browser for. They all work well for speeding up downloads and reducing distractions. They don't all provide the tools for scraping sites. Granted, there are software libraries for that, but when you have a quick task to do, some of these browsers work really well.

VERDICT

ELINKS	**8/10**	**W3M**	**6/10**
LINKS	**7/10**	**BROWSH**	**3/10**
LYNX	**6/10**		

Elinks comes top in this category because it can help you while browsing.

The Verdict
Terminal browsers

Let's be clear about one thing: you're unlikely to replace your regular browser with any of the terminal browsers on test in this month's *Roundup*. The vast majority of websites use JavaScript, and many sites will just refuse to run without it, so this is one limitation to bear in mind.

However, if you're looking for answers on forums or collecting data from sites, you're usually going to be fine with a terminal browser at your fingertips. It's a tool that's light and quick to carry out common tasks. Distractions are kept to a minimum simply because support is deliberately lacking for the worst offenders in this area: JavaScript and CSS.

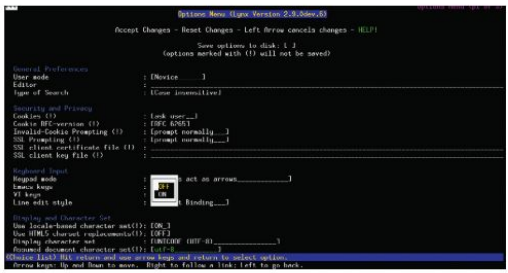
Since so many things are either turned off or not implemented, many of the methods of tracking your progress online are eliminated. To avoid tracking, you'll want to manage your cookies. The tools for this task are much more clear, and it's easier to clear cookies selectively than in the big browsers.

With much of the graphic content removed the speed of download is greatly increased. *Vim* users can use standard keybindings without going through heaps of installs and odd extensions. You can pull down media using special features so you can enjoy music and video despite cutting out most graphics.

Simply surfing by using a terminal is not the only reason to consider these browsers. They have other features that will enhance your experience with them. Through the use of scripting and calling other software you can make them even more capable. As is the Linux way, you create your own solutions by connecting the best tool for each type of content and task that you want done.

To do this, you can speed up fact-finding missions using the output options of these browsers and then combining them with parsers and presentation software. You even have the *w3m* browser that you can use as a pager for your email, directory view and comparison of files. All these capabilities makes your browser an advanced tool rather than just a text viewer.

Browsh comes last in many of this *Roundup's* categories, but don't discount it entirely. It's an outlier because it's been developed with a different set of goals in mind. While you may want to drag down all your data, sometimes you just want to explore a site properly – *browsh* enables you to do this.



1st Elinks 8/10

Web: http://elinks.or.cz Licence: GPLv2
Version: 0.13.2
Elinks can display your pages, supports more standards and also has menus.

2nd w3m 7/10

Web: http://w3m.sourceforge.net Licence: MIT
Version: 0.5.3
w3m's strength is that it can double up as a pager and image viewer.

3rd browsh 6/10

Web: www.brow.shLicence: LGPLv2
Version: 1.6.4
browsh may not have fared well in this comparison, but it looks good.

4th Links 6/10

Web: http://links.twibright.com/user_en.html Licence: GPLv2
Version: 2.21
Links can do most things well enough, but is outclassed by the other browsers.

5th Lynx 5/10

Web: https://lynx.browser.org Licence: GPLv2
Version: 2.9.0dev.6
Lynx is fast and can be scripted, but that's about it.

» ALSO CONSIDER

Having a browser in the terminal is great, but you may be in another type of software: *Emacs*. If you are, consider exploring the *eww* browser, which presents your web page right there in *Emacs*. It even show pictures, to a degree. Search engines display nicely and you can continue in your favourite program.

Downloading and saving the pages is at your fingertips and you can cut and paste as you would any other document. Cookies are handled and you can download the source in another buffer. That buffer will be in HTML mode, so highlighting works correctly.

When rendering the page, *eww* will use the colours as defined in the page you're downloading. There's no JavaScript support and it handles nested tables poorly (according to the designer), but simple documentation pages and support pages will work without issue. Staying in *Emacs* is a deal-breaker for some people. If this sounds like you, check out *Eww* ASAP.

CELEBRATE 30 YEARS OF LINUX!

How a 21-year-old's bedroom coding project took over the world and a few other things along the way.

Linux only exists because of Christmas. On January 5, 1991, a 21-year-old computer science student, who was currently living with his mum, trudged through the (we assume) snow-covered streets of Helsinki, with his pockets stuffed full of Christmas gift money. Linus Torvalds wandered up to his local PC store and purchased his first PC, an Intel 386 DX33, with 4MB of memory and a 40MB hard drive. On this stalwart machine he would write the first-ever version of Linux. From this moment on, the history of Linux becomes a love story about community collaboration, open-source development, software freedom and open platforms.

Previous to walking into that computer store, Linus Torvalds had tinkered on the obscure (UK-designed) Sinclair QL (Quantum Leap) and the far better-known Commodore VIC-20. Fine home computers, but neither was going to birth a world-straddling kernel. A boy needs standards to make something that will be adopted worldwide, and an IBM-compatible PC was a perfect place to start. But we're sure Torvalds' mind was focused more on having fun with *Prince of Persia* at that point than specifically developing a Microsoft-conquering kernel.

LINUS TORVALDS' ACHIEVEMENT

"A 21-year-old, barely able to afford an Intel 386 DX33, was about to start a development process that would support a software ecosystem..."

Let's be clear: a 21-year-old, barely able to afford an Intel 386 DX33, was about to start a development process that would support a software ecosystem, which in turn would run most of the smart devices in the world, a majority of the internet, all of the world's fastest supercomputers, chunks of Hollywood's special effects industry, SpaceX rockets, NASA Mars probes, self-driving cars, tens of millions of SBC like the Pi and a whole bunch of other stuff. How the heck did that happen? Turn the page to find out...

Pre-Linux development

Discover how Unix and GNU became the foundation of Linus Torvalds' brainchild.

To understand how Linux got started, you need to understand Unix. Before Linux, Unix was a well-established operating system standard through the 1960s into the 1970s. It was already powering mainframes built by the likes of IBM, HP, and AT&T. We're not talking small fry, then – they were mega corporations selling their products around the globe.

If we look at the development of Unix, you'll see certain parallels with Linux: freethinking academic types who were given free rein to develop what they want. But whereas Unix was ultimately boxed into closed-source corporatism, tied to a fixed and dwindling development team, eroded by profit margins and lawyers' fees,

Ken Thomas (left) and Dennis Ritchie are credited with largely creating much of the original UNIX family of operating systems, while Ritchie also created the C language.

groups that followed Linux embraced a more strict open approach. This enabled free experimentation, development and collaboration on a worldwide scale. Yeah, yeah, you get the point!

Back to Unix, which is an operating system standard that started development in academia at the end of the 1960s as part of MIT, Bell Labs and then part of AT&T. The initially single or uni-processing OS, spawned from the Multics OS, was dubbed Unics, with an assembler, editor and the B programming language. At some point that "cs" was swapped to an "x," probably because it was cooler, dude.

At some point, someone needed a text editor to run on a DEC PDP-11 machine. So, the Unix team obliged and developed *roff* and *troff*, the first digital typesetting system. Such unfettered functionality demanded documentation, so the "man" system (still used to this day) was created with the first Unix Programming Manual in November 1971. This was all a stroke of luck, because the DEC PDP-11 was the most popular mini-mainframe of its day, and everyone focused on the neatly documented and openly shared Unix system.

In 1973, version 4 of Unix was rewritten in portable C, though it would be five years until anyone tried running Unix on anything but a PDP-11. At this point, a copy of the Unix source code cost almost $100,000 in current money to licence from AT&T, so commercial use was limited during the 70s. However, by the early 80s costs had rapidly dropped and widespread use at Bell Labs, AT&T, and among computer science students propelled the use of Unix. It was considered a universal OS standard, and in the mid-1980s the POSIX standard was proposed by the IEEE, backed by the US government. This makes any operating system following POSIX at

» LINUX RUNS EVERYTHING

Developing software for supercomputers is expensive. During the 1980s, Cray was spending as much on software development as it was on its hardware. In a trend that would only grow, Cray initially shifted to UNIX System V, then a BSD-based OS, and eventually, in 2004, SUSE Linux to power its supercomputers. This was matched across the sector, and the top 500 supercomputers (**www.top500.org**) now all run Linux.

Internet services have also all been developed to run on Unix systems. Microsoft and BSD systems do retain a good slice of services, but over 50 per cent of web servers are powered by Linux. Recent moves to virtual services with container-based deployment are all Linux-based. Microsoft's cloud service Azure reports that Linux is its largest deployment OS and, more to the point, Google uses Linux to power most of its services, as do many other service suppliers aka AWS.

Android's mobile OS share dropped in 2020 to just 84 per cent – it's powered by Linux. Google bought the startup that was developing Android in 2005. LineageOS (**https://lineageos.org**) is a well-maintained fork of Android and supports most popular handsets well after their manufacturers abandon them.

Space was thought to be Linux's final frontier, because it's not a certified deterministic OS, which is the gold standard for real-time OSes in mission-critical situations. Turns out that SpaceX rockets use Linux to power their flight systems, using a triple-redundancy system, while NASA has sent Linux to Mars in its helicopter drone, Ingenuity. Tesla is also reportedly running Linux in its cars.

Linux has also been at the heart of Hollywood's special effects since 1997's *Titanic* used a Linux server farm of DEC Alphas at Digital Domain to create its CGI. DreamWorks' *Shrek* in 2001 was the first film that was entirely created on Linux systems. Meanwhile, Pixar ported its Renderman system to Linux from SGI and Sun servers around 2000, in time to produce *Finding Nemo* in 2003.

Linus Tovalds being interviewed by Linux Format back in 2012.

least partly if not largely compatible with other versions.

At the end of the 1980s, the Unix story got messy, with commercial infighting, competing standards and closing off of standards, often dubbed Unix Wars. While AT&T, Sun Microsystems, Oracle, SCO, and others argued, a Finnish boy was about to start university...

We GNU that

Before we dive into the early world of Linux, there's another part of the puzzle of its success that we need to put in place: the GNU Project, established by Richard Stallman. Stallman was a product of the 1970s development environment: a freethinking, academic, hippy type. One day, he couldn't use a printer, and because the company refused to supply the source code, he couldn't fix the issue – supplying source code was quite normal at the time. He went apoplectic and established a free software development revolution: an entire free OS ecosystem, free software licence and philosophy that's still going strong. Take that, proprietary software!

The GNU Project was established by Stallman in 1983, with GNU being a hilarious (to hackers, at least) recursive acronym for "GNU is Not Unix." Geddit? Its aim was to establish a free OS ecosystem with all the tools and services a fully functioning OS requires. Do keep in mind that most of the tools created then are still being used and maintained today.

By 1987, GNU had established its own compiler, *GCC*, the *Emacs* editor, the basis of the GNU Core Utilities (basic file manipulation tools such as list, copy, delete and so on), a rudimentary kernel and a chess engine (*See* **LXF273**). But more importantly, Stallman had cemented his ideal of software freedom with the 1989 "copyleft" GPL software licence, and his manifesto setting out the four software freedoms enabling users to run, study, modify and distribute any software – including the source – for any reason.

> **RICHARD STALLMAN'S GNU WORK**
> "He established a free software development revolution: an entire free OS ecosystem, free software licence and philosophy that's still going strong."

Linux Format interviewed Richard Stallman, the creator of the GNU free software movement, in 2011.

The GPL remains the strongest copyleft licence, and while it has perhaps fallen out of vogue, it's still regarded as the best licence for true open-source development, and cements most Linux distros. *GCC* is still an industry standard, *Emacs* remains a feature-rich development environment, and the GNU Core Utilities are still widely used in certain POSIX systems and most Linux distros.

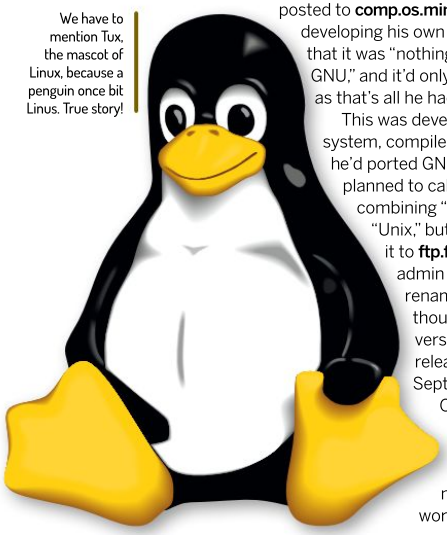
You could argue that without the GNU Project being established, Linux would never have taken off. The GPL licence (adopted early on in Linux development) forces all developers to share back their enhancements to the source code. It's a feedback loop that promotes shared improvements. Alternative open-source licences enable corporations to take source code and never share back

improvements, meaning the base code is more likely to remain static. This was backed by a generation of developers that grew up studying and using Unix, looking to contribute to a truly freed open-source OS.

Let's all Freax out!

We're getting ahead of ourselves. Linus Torvalds had his Intel 386, was studying computer science at the University of Helsinki, and was using the MINIX 16-bit OS and kernel. MINIX is a POSIX-compatible Unix-like OS and micro-kernel. In 1991, it had a liberal licence, costing just $69, offering the source code but restricted modification and redistribution.

We imagine the 16-bit limitation spurred Torvalds to create his own 32-bit kernel, but he states the licence restrictions were also key. So, on 25 August, 1991, he posted to **comp.os.minix** that he was developing his own free OS. He said that it was "nothing professional like GNU," and it'd only support AT disks, as that's all he had.



We have to mention Tux, the mascot of Linux, because a penguin once bit Linus. True story!

This was developed on a MINIX system, compiled on GNU *GCC*, and he'd ported GNU *bash*. Torvalds had planned to call his OS Freax, combining "Free," "Freak," and "Unix," but once he'd uploaded it to **ftp.funet.fi**, a volunteer admin (Ari Lemmke) renamed it Linux, as he thought it was better. So, version 0.01 of Linux was released to the world in September 1991.

One telling part of the release notes states: "A kernel by itself gets you nowhere. To get a working system you need a shell, compilers, a library, etc... Most of the tools used with Linux are GNU software and are under the GNU copyleft. These tools aren't in the distribution – ask me (or GNU) for more info."

Minix for all of its creator's protestations to its superiority has ceased development, even though it runs Intel's CPU Management Engine.

Importantly, this outlines Linux's reliance on other GPL-licenced tools, and shows the use of the term "distribution," now shortened to "distro." As Torvalds points out, an operating system isn't a kernel alone; it's a collection of tools, scripts, configs, drivers, services and a kernel, lumped together in an easier form for users to install and use.

As for the licence, Torvalds initially used his own, which restricted commercial use, but by January 1992 he'd been asked to adopt the GPL, and had stated the kernel licence would change to align it with the other tools being used. It was December 1992, and for the release of v0.99, the Linux kernel was GPLv2-licenced. This cemented the legal clause that anyone using the kernel source has to contribute back any changes used in published code.

» BIRTH OF THE LINUX FOUNDATION

Open Source Development Labs was set up at the turn of the millennium to, among other things, get Linux into data centres and communication networks. They became Torvald's (and his right-hand man Andrew Morton's) employer in 2003. Prior to this he was employed by Transmeta, who permitted him to continue kernel development alongside his other work. Five years previous, another consortium, the Free Standards Group had been set up. By 2007 its work was mostly driving people to switch to Linux, and the two groups merged to form the Linux Foundation (LF).

Today the LF's Platinum members include Facebook, Microsoft, Tencent, IBM and Intel. All of whom, contribute (besides the half a million dollars required for Platinum status) a great deal of code to the Kernel. In 2012, when Microsoft wanted to get Linux working on its Azure cloud, they were for a time the biggest contributor. Besides funding the Kernel, the LF host hundreds of other open source projects, including Let's Encrypt, the OpenJS Foundation and the Core Infrastructure Initiative, which aims to secure the software which underpins the internet.

But it's not all code and corporations. There's conferences too, and it's thanks to the Linux Foundation that we've been able to provide interviews and coverage from the annual Open Source Summit. We look forward to conference season resuming, so we can get back to the snack bars and coffee counters.

Early kernel development

Refining the very heart of Linux hasn't been an easy ride over the years...

Are you a recent Linux convert who's had to engage in combat with rogue configuration files, misbehaving drivers or other baffling failures? Then spare a thought for those early adopters whose bug reports and invective utterances blazed the trail for contemporary desktop Linux. Up until comparatively recently, it was entirely possible to destroy your monitor by feeding X invalid timing information. Ever had problems with Grub? Try fighting it out with an early version of Lilo.

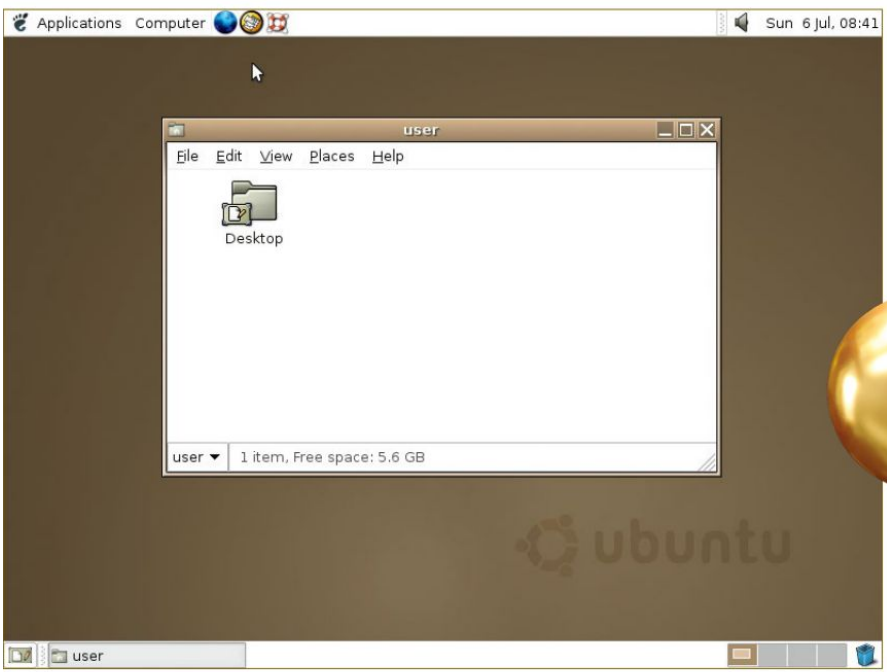
In the early days, even getting a mouse to work was non-trivial, requiring the user to do all kinds of manual calibration. Red Hat released a tool called *Xconfigurator* that provided a text-mode, menu-driven interface for setting up the X server. It was considered a godsend, even though all it did was generate an **XF86Config** file which otherwise you'd have to write yourself.

So while in 2000 users whined about Windows Me being slow and disabling real-mode DOS, your average Linux user would jump for joy if their installation process completed. Even if you got to that stage, it would be foolishly optimistic to suppose the OS would boot successfully. Hardware detection was virtually non-existent, and of the few drivers that had been written for Linux, most weren't production quality. Yet somehow, the pioneers persisted – many were of the mindset that preferred the DOS way of working, which began to be sidelined as the millennium approached. Windows users were having their files abstracted away – 'My Computer' epitomises this movement.

In January 2001 Kernel 2.4 was released and with it came support for USB and the exciting new Pentium IV processors, among other things. It was of particular importance to desktop users thanks to its unified treatment of PCI, ISA, PC Card and PnP devices as well as ACPI support. The dot-com bubble was just about to burst, but all the excitement and speculation around it meant that many computer enthusiasts had a broadband connection in their home, some even enjoyed the luxury of owning more than one computer.

User-unfriendly Linux

This solved some major entry barriers to Linux: people could now download it much more easily; up-to-date documentation was easily accessible; and when Linux saw fit to disappear one's internet connection (or render the system unbootable), the other machine could be used to seek guidance. But the user experience was still, on the whole, woefully inhospitable. While some installers had evolved graphical capabilities, these more often than not were more trouble than they were worth. Users were expected to understand the ins and outs of



The Human theme was an attempt to make Ubuntu Linux more friendly, because as everyone knows brown is beautiful, especially if you're a warthog.

disk partitioning, and be able to discern which packages they required from often terse descriptions.

Windows XP was released in October 2001, and while this was seen as a vast improvement over its predecessor, many users found that their machines weren't up to running it. After all, it required 64MB RAM and a whopping 1.5GB of disk space. Remember that BIOSes had only recently gained the ability to address large drives (there were various limits, depending on the BIOS, 2.1, 4.2 and 8.4GB were common barriers). So many people couldn't install it on their hardware, and many that met the minimum specs found the performance rapidly degraded once the usual pantheon of office suites and runtime libraries were installed.

This provided the motivation for another minor exodus to Linux, and the retro-hardware contingent continue to make up a key part of the Linux userbase (and berate us for not including 32-bit distros). Before 2006 all Macs had PowerPC processors, and many of these (as well as early Intel Macs), long-bereft of software updates from Apple, now run Linux too.

Gnome makes an appearance

The Gnome 2 desktop environment was released in 2002 and this would become a desktop so influential that some still seek (whether out of nostalgia, atavism or curmudgeonly dislike of modern alternatives) to reproduce it. It aimed to be as simple, tweakable and intuitive, and it's hard to argue against its achieving all of these adjectives. One of the major enablers was its strict adherence to the Gnome Human Interface Guidelines, which set out some key principles for application designers. This meant the desktop was consistent not just internally, but in respect to all the GTK apps that people would go on to write for it.

Also released was KDE 3, which vaguely resembled Windows – in that it was cosmetically similar and slightly more resource-demanding than Gnome. People and distributions sided with one or the other. SUSE Linux (predecessor of openSUSE) always aimed to be desktop agnostic, but went KDE-only in 2009. Today it caters to both Gnome and KDE.

In late 2002, 'DVD' Jon Johansen was charged over the 1999 release of the *DeCSS* software for circumventing the Content Scrambling System (CSS) used on commercial DVDs. This software enabled Linux users to play DVDs, a feat they had been hitherto unable to do since DVD software required a licence key from the DVD Copy Control Agency, one of the plaintiffs in the suit. It later emerged that CSS could be broken much more trivially and Johansen was eventually acquitted. By this time iPods and piracy meant that MP3 files were commonplace. These were dogged by patent issues with a number of bodies asserting ownership of various parts of the underlying algorithm. As a result, many distros shipped without patent-

» BIG BUSINESS VS LINUX

Being the root of all evil, whenever money is involved, things can turn nasty. So, when the big players in the enterprise and business markets began to see Linux distros as a threat, lawyers were called.

A series of leaked Microsoft memos from August 1998, known as the Halloween Documents for the date they were released, detailed Microsoft's private worries that Linux, and open-source development in general, was a direct threat to its business, along with ways to combat its uptake. This private view was in direct conflict with the company's public line on the matter, though Steve Ballmer infamously called Linux a cancer in 2001. The documents are available at **www.catb.org/~esr/halloween**, and in them Microsoft predicted that "Linux is on track to eventually own the x86 UNIX market..." It was correct.

There was little Microsoft could do to combat Linux, as it couldn't be bought. The documents suggested extending open protocols with Microsoft's own proprietary extensions (that didn't work), and seeding the market with fear, uncertainty and doubt (FUD) also failed.

There was another angle, however: help a company that's suing over copyright infringement of the source code. In 2003, a company called SCO claimed part of its UNIX System V source code was being used within Linux, making it an unauthorised derivative of UNIX. SCO sued IBM for $1 billion (among many other companies), and demanded end users pay a Linux licence fee. Microsoft leaped into action and paid SCO $106 million, as detailed in a leaked and verified SCO memo. After years of legal arguments, a code audit found there to be no evidence of copied UNIX code in the Linux kernel. SCO went bankrupt in 2009, but parts of the lawsuit still rumble on.

Timeline

25 AUGUST 1991

Linus announces on comp.os.minix
Linus Torvalds, a 21-year-old student at the University of Helsinki, Finland, starts toying with the idea of creating his own clone of the Minix OS.

17 SEPTEMBER 1991

v0.01 Posted on ftp.funet.fi
This release includes Bash v1.08 and GCC v1.40. At this time, the source-only OS is free of any Minix code and has a multi-threaded file system.

NOVEMBER 1991

v0.10 Linux is self-building
Linus overwrites critical parts of his Minix partition. Since he couldn't boot into Minix, he decided to write the programs to compile Linux under itself.

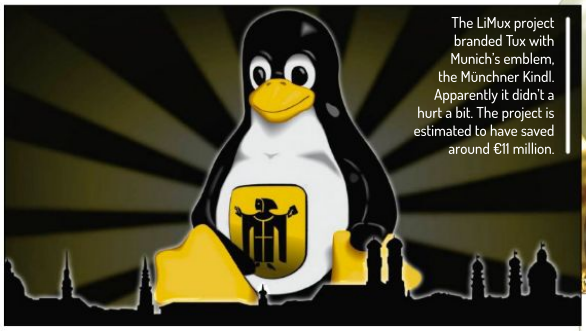
5 JANUARY 1992

v0.12 GPL licenced
Linux originally had its own licence to restrict commercial activity. Linus switches to GPL with this release.

1991
Python

May 1992
Softlanding Linux System

encumbered multimedia codecs. The law is murky though, and rights holders have shown restraint in filing suit against FOSS implementations of these codecs. Most distros are prudent and leave it up to the user to install these, although Ubuntu and derivatives will do so if you tick a box. The MP3 patent expired in 2017, though it doesn't really matter – we have plenty of open formats and codecs now (OGG, FLAC, VPx and x264). It's still technically a DMCA violation to use *libdvdcss* (a modern and much more efficient way of cracking CSS, used by the majority of media players on Linux) to watch a DVD, but that only applies in some countries and to date, no one has challenged its use.



The LiMux project branded Tux with Munich's emblem, the Münchner Kindl. Apparently it didn't a hurt a bit. The project is estimated to have saved around €11 million.

Early kernel development

As Linux gained traction, first among academics and hobbyists and then, by the mid-90s, when businesses started to form around it, the number of contributors bloomed. One take from Linus himself, is that once the X Windows System was working on Linux (with v0.95) it became much more appealing. So one could infer that even in 1992 people were afraid of the command line. This popularity led to the establishment of the maintainer heirarchy so that patches submitted could be reviewed and promoted efficiently to Linus' source tree. Though that first version of the **MAINTAINERS** file describes Linus as "buried alive in email".

The email-centric development process is still followed today, except that the Official Linux Kernel Mailing List was set up in 1997, and now Git is used for version control. So it's a lot easier to make sure you're working on an up-to-date branch, rather than having to wait for floppies in the mail. Patches are still generated using `diff -u` to show which lines have been changed in which files. Before Git, the proprietary *BitKeeper* concurrent versioning system (CVS) was used. And when this arrangement came to an end (helped by Andrew Tridge's reverse engineering mischief), Torvalds got hacking and 10 days later there was Git.

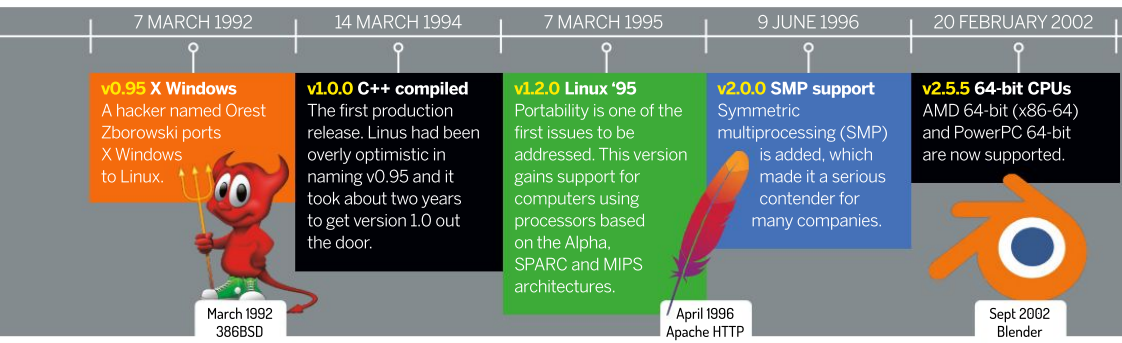
After two years in development Kernel 2.6 was released in 2003. This was a vastly different beast to 2.4, featuring scheduler enhancements, improved support for multiprocessor systems (including hyperthreading, NPTL and NUMA support), faster I/O and a huge amount of extra hardware support. We also saw the Physical Address Extension (PAE) so that 32-bit machines could address up to 64GB of RAM (before they were limited to about 3.2GB).

Also introduced was the venerable Advanced Linux Sound Architecture (ALSA) subsystem. This enabled (almost) out-of-the-box functionality for popular sound cards, as well as support for multiple devices, hardware mixing, full-duplex operation and MIDI.

The most far-reaching new feature was the old device management subsystem, devfs, being superceded by udev. This didn't appear until 2.6.13 (November 2003), at which point the **/dev** directory ceased to be a list of (many, many) static nodes and became a dynamic reflection of the devices actually connected to the system. The subsystem udev also handled firmware loading, and userspace events and contributed to a much more convenient experience for desktop users. Although you still relied on such arcana as HAL and ivman in order to automount a USB stick with the correct permissions.

> **SOUND FEATURES IN KERNEL 2.6**
> "The venerable Advanced Linux Sound Architecture (ALSA) subsystem enabled (almost) out-of-the-box functionality for popular sound cards."

Linux (having already been ported to non-x86 64 bit processors) supported the Itanium's IA64 instruction when it was released in 2001. This architecture was doomed to fail though, and Intel eventually moved to the more conservative AMD64 (or x86-64) architecture,



which has been around since 2003. Thanks to open source development, Linux users were running 64-bit desktops right away, while Windows users would have to wait until 2005 for the x64 release of XP. Various proprietary applications (notably Steam and lots its games) run in 32-bit mode, which provides some motivation for distributions to maintain at least some 32-bit libraries. Debian 11 will support 32-bit x86 in some form until 2026, but most other distros have abandoned it. Eventually such machines will go the way of the 386, no longer supported on Linux since 2013.

Enter the archetype

In 2004, a sound server called *Polypaudio* was released by a hitherto unknown developer called Lennart Poettering and some others. At this time desktop environments relied on sound servers to overcome shortcomings in ALSA's dmix system: Gnome was using the *Enlightened Sound Daemon* (*ESD*) and KDE was using the analogue *Realtime* synthesizer (*aRts*). *Polypaudio* was designed to be a drop-in replacement for *ESD*, providing much more advanced features, such as per-application volume control and network transparency. In 2006 the project, citing criticism that nobody wants polyps, renamed itself *PulseAudio* (it was in fact named after the sea-dwelling creature).

With its new name and increased demand for a sound system comparable with that of OSX or the newly released (and much maligned) Windows Vista, *PulseAudio* enjoyed substantial development and began to be considered for inclusion in many distros. As is traditional, Fedora was the first to adopt, incorporating it as the default in version 8, released in late 2007.

Asus' EeePC Linux was based on Xandros and IceWM, but beginners didn't like it, and professionals just replaced it.

Ubuntu followed suit in 8.04, although its implementation attracted much criticism and resulted in much anti-*Pulse* vitriol. Poettering at one stage even described his brainchild as "the software that currently breaks your audio." It took some time but eventually Ubuntu (and other distros) sorted out implementation issues, and it mostly worked out of the box. Now we have Pipewire in the works for a new generation of audio-based rage against the machine.

The cost of progress

The year 2010 may be remembered by some as the one Ubuntu started to lose the plot. Its Ubuntu *Software Center* now included paid-for apps and the Netbook remix used a new desktop called Unity. In the 11.04 release though, this became the new shell for the main release too. Ubuntu had long taken issue with the new Gnome 3 desktop, which at the time of the Ubuntu feature-freeze was not considered stable enough to include in the release anyway, and Gnome 2 was already a relic. So in a sense Ubuntu had no choice, but no one likes change, and users were quick to bemoan the new desktops. Of course things have come full circle with Ubuntu using Gnome 3 once more since 20.04 and people bemoaning the loss of Unity.

Gnome 3 is not without controversy too. First, many preferred the old Gnome 2 way of doing things and this clearly was not that. Second, all the fancy desktop effects required a reasonable graphics card (and also working drivers). There was a fallback mode, but it severely crippled desktop usability. Finally, this appeared to be something designed for use on mobiles or tablets, yet even today mobile Linux (not counting Android) has never taken off, so why should users be forced into this mode of thinking? Many found though, that once some old habits are unlearned and some sneaky keyboard shortcuts are learned (and Gnome's *Tweak Tool* is installed), that the Gnome 3 way of working could be just as efficient, if not more so, than its predecessor. KDE users looked on smugly, having already gone through all the rigmarole of desktop modernisation (albeit less drastic than Gnome's) when KDE 4 was released in 2008.

Around this point we ought to mention Systemd as well, but there's not much to say that hasn't been said elsewhere: the old init system was creaking at the seams; a new and better one came along; it wasn't everyone's cup of tea, but we use it anyway; the internet slanders Lennart Poettering.

Timeline

17 DECEMBER 2003

v2.6.0 The Beaver Detox
Major overhaul to Loadable Kernel Modules (LKM). Improved performance for enterprise-class hardware, the Virtual Memory subsystem, the CPU scheduler and the I/O scheduler.

April 2003
Firefox

29 NOVEMBER 2006

v2.6.19 ext4
Experimental support for the ext4 filesystem.

August 2006
Linux Mint

5 FEBRUARY 2007

v2.6.20 KVM arrives
Kernel-based Virtual Machine (KVM) is merged, adding Intel and AMD hardware virtualisation extensions.

25 DECEMBER 2008

v2.6.28 Graphics rewrite
The Linux graphics stack was fully updated to ensure that it utilised the full power of modern GPUs.

September 2008
Chromium

Distro developments

A single kernel has enabled a good number of Linux distributions to blossom into life.

After looking into the development of the Linux kernel itself and the surrounding supporting software, let's turn to how Linux distributions (distros) from this point were developed and branched into a wide-ranging ecosystem.

Distros enabled the use of the Linux kernel to grow rapidly. Not only did they ease the installation of Linux (which early on was a complex process of source compilation, gathering the right tools, creating filesystem layouts by hand, and bootloaders, all from the terminal on systems with limited resources), but one distro can also become the base for a whole new distro, tailored for a new use or audience.

Primordial soup

As Linux v0.01 was only released in September 1991, the first distribution of Linux – though by modern standards, it's lacking in every department – created by HJ Lu, was simply called Linux 0.12. Released at the end of 1991, it came on two 5.25-inch floppy disks, and required a HEX editor to get running. One disk was a kernel boot disk, the other stored the root OS tools.

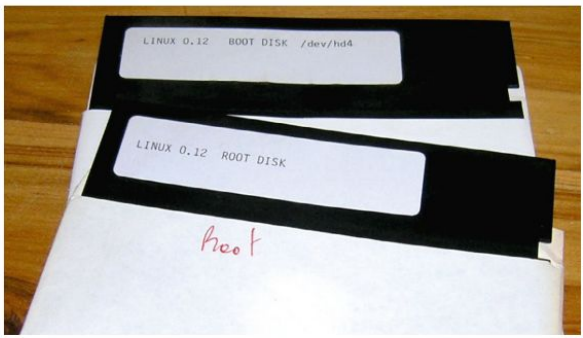
In those early days of distro evolution, things changed rapidly. Development was quickly adding base functionality, and people were trying out the best ways to package a Linux-based OS. MCC Interim Linux was released in February 1992 with an improved text-based installer, and was made available through an FTP server.

X Windows – the standard Unix windowing system – was ported, and TAMU Linux was released in May 1992 with it packaged: making it the first graphical distro.

While all of these are notable as being among the first Linux distros, they didn't last. The same can be said for Softlanding Linux System (SLS), also released in May 1992, which packaged X Windows and a TCP/IP network stack. It's notable, though, because of its shortcomings (bugs and a change to the executable system) inspired the creation of the two longest-running and, in many ways, most influential Linux distros: Slackware and Debian.

Nowadays, a number of base distros appear, reliably maintained by individuals, groups, or businesses. Once they're established, stable and become popular, offshoots branch from these root distros offering new specialisations or features. This creates a number of base distro genera, formed around the original package manager and software repositories.

The effect is a Linux family tree (*see page 43*), where you can date all distros back to an initial root release. Some branches sprout and die; either the group maintaining it disbands or there's no wider interest. Some branches become so popular they create a whole new genus, becoming the basis for a further expansion.



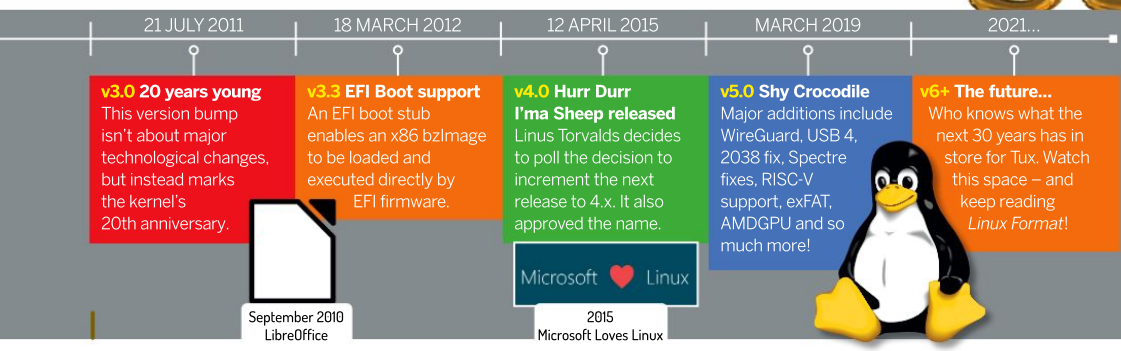
The first Linux distro, aptly named: Linux 0.12.

CREDIT: Shermozle, CC BY-SA 3.0 https://commons.wikimedia.org/wiki/File:Linux_0_12.jpg

Evolution, not revolution

As with plants and animals, offshoots inherit traits, the base install, package manager, and software repositories being key. A package manager is how the OS installs, updates, removes and maintains the installed software, which includes downloading software packages from the managed software servers, called repositories. This can become contentious – these child distros are leeching off the parent's bandwidth – but initially, while they're growing, this use won't look much different from normal user activity.

Bear in mind we're back in 1992. You're lucky if



there's a 14.4Kb/s dial-up modem at home; expensive T1 lines (1.54Mb/s) are limited to academic institutions and larger businesses. The early TAMU v1.0 distro required 18 disks for the 26MB binaries, and 35 disks for the 50MB compressed (200MB uncompressed) source code. This obviously limited access in these early days to academics and those in suitable businesses, so distro evolution was slow.

Meet the ancestors

Softlanding Linux System was popular, but it was buggy and badly maintained, so in July 1993, Patrick Volkerding forked SLS and created Slackware – so named because it wasn't a serious undertaking at the time, and was a reference to the Church of the SubGenius. This is the oldest Linux distro still maintained, and it's about to see its version 15 release after 28 years. Slackware is interesting because it's very much controlled and maintained by Volkerding, while followed by a small but enthusiastic band of users and contributors. Whereas many other distros have taken on modern enhancements, Volkerding sticks to older more traditional "Unix" ways of controlling services on Slackware. There's no formal bug tracking, no official way to contribute to the project, and no public code repository. This all makes Slackware very much an oddity that stands on its own in the Linux world. Due to its longevity, however, Slackware has attracted a couple of dozen offshoots, and at least half are still maintained today.

Thankfully, by being buggy SoftLandingLinux kickstarted some good distros!

CREDIT: Linuxcenter.ru

In August 1993, Ian Murdock, also frustrated by Softlanding Linux System, established Debian, a combination of "Debby," his girlfriend's name at the time, and "Ian." From the outset, it was established as a formal, collaborative open project in the spirit of Linux and GNU.

Early on in the Debian project, Bruce Perens maintained the base system. He went on to draft a social contract for the project and created Software in the Public Interest, a legal umbrella group to enable Debian to accept contributions. At the time, Perens was working at Pixar, so all Debian development builds are named after Toy Story characters. The Debian logo also has a strong similarity to the mark on Buzz Lightyear's chin.

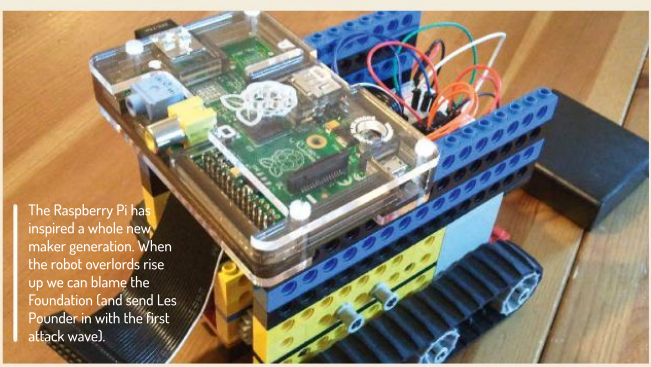
Debian is arguably the single most influential and important Linux distro ever. Just the sheer number of branches of distros from it would attest to that, but Debian is renowned for its stability, high level of testing, dedication to software freedom, and being a rigorously well-run organisation. It's testament to its creator, Ian Murdock, who sadly passed away in December 2015.

Things were still moving slowly into 1994 – there was

» THE RASPBERRY Pi SENSATION

The Raspberry Pi was released in 2012. Inspired in part by the success of the BBC Micro (hence the monogram model names) in the early 1980s, the Raspberry Pi aimed to bring practical computer science to the classrooms and bootstrap the UK electronics industry. It was only ever expected to have been produced in the thousands. Of course when it was launched, Linux was the de facto OS of choice.

While many of these devices are now empowering young coders, a great deal have become part of diverse man-cave projects: The 30-somethings who cut their teeth on BBCs, Spectrums and Commodore 64s are reliving and reviving the thrills at the interface of coding and creativity. The Raspberry Pi's GPIO pins mean that all manner of add-ons have been developed, so that the pint-sized computer can power anything from robots to remote watering systems.



The Raspberry Pi has inspired a whole new maker generation. When the robot overlords rise up we can blame the Foundation (and send Les Pounder in with the first attack wave).

The lingua franca of Pi projects is Python which, like Basic, is easy to learn. Unlike Basic, though, it's consistent, extensible and won't need to be unlearned should users move on to more advanced languages.

The Pi's support for 3D graphics is impressive, but CPU-wise it's more limited. The original Pis struggle to function as a desktop computer, even with the modest Raspbian distribution (although recent work on the *Epiphany* web browser has improved this).

In 2015 the Pi received the Pi 2 reboot, gaining a quad-core processor and extra RAM, and yet still only cost £25. Jump forward six years and we have the Pi 4 in its various forms including a full-desktop capable 8GB version the Pi 400, a range of industry-friendly models and over 30 million sales. Splendid.

just a single Slackware fork called SUSE and a few random Linux sprouts appeared, but all died out. Then in October 1994, Red Hat Linux was publicly released. Red Hat was established as a for-profit Linux business, initially selling the Red Hat Linux distribution and going on to provide support services. Red Hat went public in 1999, achieving the eighth biggest first-day gain in the history of Wall Street. It entered the NASDAQ-100 in December 2005 and topped $1 billion annual revenue in 2012. IBM purchased Red Hat in October 2018 – 24 years after its first release – for $34 billion. So that worked out very well.

A tale of hats and forks

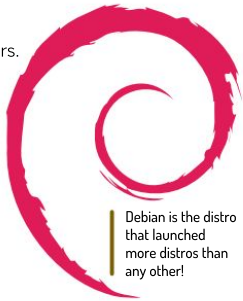
Red Hat Linux was relaunched as Red Hat Enterprise in 2001, and its commercial success attracted a wide range of forks. Notably, Red Hat directly supports Fedora as its testing distro and CentOS as its free community edition. Or it did. CentOS is being shuttered – to understandable community disdain – and a rolling release, CentOS Stream, is replacing it. As an alternative, Red Hat Enterprise is now offered freely to

The late Ian Murdock founded the influential Linux distribution Debian in 1993. Linux Format spent time talking with him in 2007.

community projects with fewer than 16 servers.

Meanwhile in Germany, SUSE (Software und System Entwicklung) started life as a commercially sold German translation of Slackware in late 1992. In 1996, an entire new SUSE distro and business was launched, based on the Dutch Jurix Linux, selling the new distro and support services.

SUSE was purchased by Novell in 2003, and in 2005, the openSUSE community edition was launched, while SUSE Linux Enterprise was developed in tandem for its commercial arm. SUSE was acquired in 2018 for $2.5 billion and returned double-digit growth through 2020, with a revenue of over $450 million. Yet despite its success, SUSE and openSUSE have only ever attracted a couple of forks. We could be wrong when we say this is



Debian is the distro that launched more distros than any other!

IAN MURDOCK'S LEGACY

"Debian is renowned for its stability, high level of testing, dedication to software freedom, and being a rigorously well-run organisation."

possibly down to their European roots.

It's a distro inferno

Between the creation of Red Hat in 1994 and 2000, there were a number of Red Hat spin-offs, because at that point there was clear commercial interest in Linux. Throughout this period, Linux was best suited to business server tasks, where much of the open-source Unix work had been focused. However, by the end of the 1990s, 56k modems had become commonplace, early home broadband was just appearing, and modern graphical desktops were in development. Linux was about to get a whole new audience.

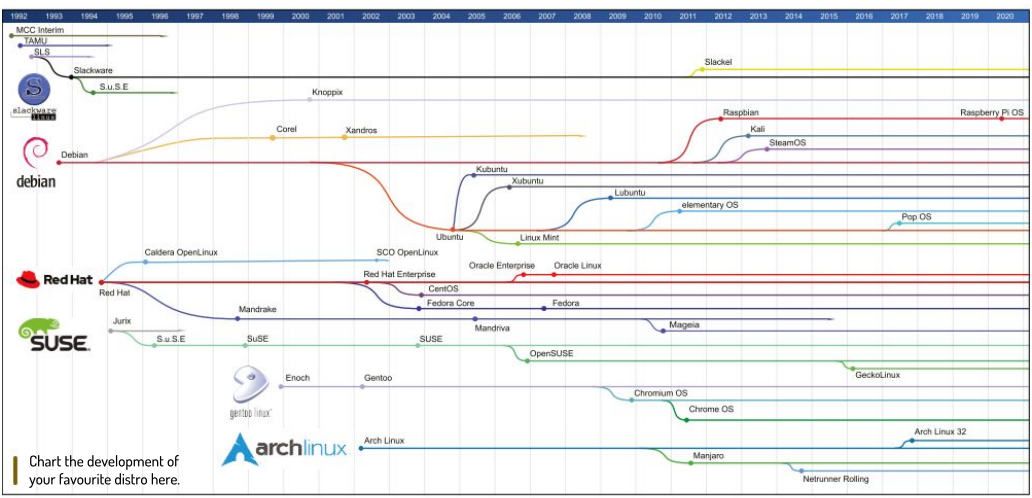


Chart the development of your favourite distro here.

CREDIT: Based on the LinuxTimeLine, by fabiololix, GNU Free Documentation License v1.3, https://github.com/FabioLolix/LinuxTimeline/tree/master

One early example was Mandrake Linux, in mid-1998. A fork of Red Hat, it was crazily aimed at making Linux easy to use for new users, using the new Kool Desktop Environment (KDE). The French/Brazilian development team gained a lot of attention but, ultimately, financial problems closed the project in 2011. However, its spirit continues in the excellent but less well-known Mageia and OpenMandriva projects.

A distro with humanity in mind

With Mandrake pointing the way, the early 2000s saw an explosion of distro releases. Now that the Debian project at this point was well established, well regarded and well known, it became the basis for hundreds of Linux distros. But we'll only mention one: Ubuntu, released in 2004 by South African millionaire Mark Shuttleworth, who jokingly calls himself the self-appointed benevolent dictator for life. The Ubuntu Foundation was created in 2005 as a philanthropic project – Ubuntu is a Zulu word meaning humanity – to provide quality open-source software, with Canonical as the supporting commercial arm.

With big bucks, comes big offices! Here's the Red Hat HQ sporting its old logo.

CREDIT: Bz3rk, CC BY-SA 3.0 https://en.wikipedia.org/wiki/Red_Hat#/media/File:Red_Hat_headquarters_at_Raleigh,_North_Carolina,_US_--_9_November_2013.jpg

Ubuntu as a branch of Debian has itself seen over 80 distros fork from it, while Ubuntu has the highest share of all desktop Linux installs – though this is notoriously hard to measure – when users are polled. Why Ubuntu became so popular is hard to fully pinpoint. Key is just like Mandrake before it, Ubuntu set out to make desktop Linux easy for first-time users. It also offered the distro on free CDs via its ShipIt service until 2011, alongside fast, reliable server downloads. Furthermore, it was based on the popular Debian, it jumped on the new, slick Gnome desktop, and it set out a regular six-month release cycle, with a Long Term Support release every two years. Support was for 18 months (now nine months) for regular releases, and 36 months for LTS ones (now five years).

Ubuntu also offered great forums and help sites, along with a community council, and support for forks such as Xubuntu, Lubuntu and many others. It had sane defaults, too, and made it easier to install display drivers (an absolute pain 10-plus years ago), while offering a huge catalogue of tested, ready-to-run open-source software and dedicated server builds. We guess when you say all this out loud, it sounds pretty compelling!

Two core release branches we'll quickly mention are Arch Linux and Gentoo, both released around 2000. Gentoo (named after the fastest penguin in the world) is a built-from-source distro compiled with specific optimisations for the hardware it's going to run on. This is very clever, but also very time-consuming. Google Chrome OS is derived from Gentoo. In early 2002, Arch Linux was released, devised as a minimalist distro, where the user does much of the installation work to create an OS with just the parts required. This DIY approach was partly why Arch is renowned for its amazing documentation and for rolling out the earliest release of new versions of software.

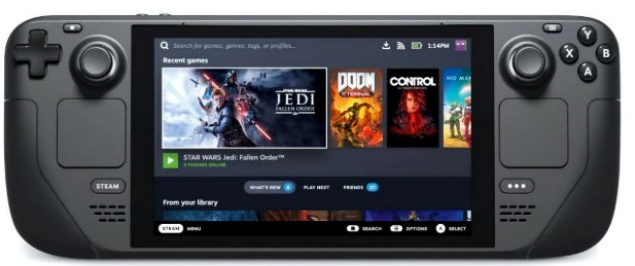
» GET YOUR LINUX GAME ON

There's always been a niche interest in gaming on Linux, but this was mostly done through *Wine*, which has been around since the mid-90s and frankly always felt like a sticking plaster to enable *World of Warcraft* or whatever the current Windows game of choice was to be played on Linux.

Things started to changed when Valve ported its Source engine to Linux along with releasing its Steam for Linux client in 2012. This opened the gate for Source-based native Linux game distribution. In addition, at the end of 2013 Valve announced it was creating SteamOS a dedicated Debian-based distro for running its Steam client. This was to tie in later with its failed attempt at creating a Steam Machine ecosystem. Today there are over 7,000 native Linux games available on Steam, out of around 14,000 in total.

Perhaps more significantly is that Valve never stopped developing SteamOS, despite its Steam Machine failure. In 2018 Valve released its own internal folk of *Wine* called *Proton* that was integrated into Steam itself and propelled Linux support for Windows games to a new level, with currently a reported 50 per cent of games offering Platinum compatibility.

But why all this work just to help one per cent of Steam's Linux-using gamers? This summer Valve revealed its Steam Deck, a Linux-powered hand-held PC console, which it promised would run all Windows games via its Steam Proton layer. Perhaps 2021 is year of the Linux desktop after all...



Thanks to Steam on Linux, Tux gamers finally have thousands of games to play, and Linux Format writers can peruse the Summer Sale offerings and still claim to be doing work.

Google's Android (not a distro) is frowned upon in the Linux world, but you can't deny the effect it had on the market.

At the height of the distro madness (around 2010), there were almost 300 Linux distros, we'd argue an unsustainable number, with many just repeating basic desktop functionality already available in core root distros. Progressing into the 2000s, and with increasing complexity in maintaining a modern OS, the number of Linux distros started to reduce, but that didn't stop well-organised groups creating popular new distro forks when they felt a need.

A good example is Raspberry Pi OS, a rebrand of Raspbian, itself a fork of Debian. The new Arm-based hardware platform needed a dedicated operating system, so picking up Debian and refitting it for the Raspberry Pi, including educational software, libraries for its GPIO access, and tailored tools to configure its hardware, made absolute sense.

Linux hardware specialist System76 was tired of niggling software issues associated with using other distros, and wanted direct control. So, it introduced Pop!_OS, a fork of Ubuntu, to not only directly support its laptops and desktop hardware, but also its customers' needs. It's a slick, modern distro, with support for popular software and hardware.

Linux Mint started in 2006 as a small personal Ubuntu fork project. When Ubuntu changed to its "modern" Unity desktop design in 2011, many users revolted. The Linux Mint project created its own "classic" desktop, called Cinnamon, in 2012, and it brought many former Ubuntu users with it. The Linux Mint project has stuck with its "user first" design approach, and evolved remarkably well.

This doesn't even touch upon commercially focused distros, such as Android, Chrome OS, Intel's ClearOS, Google's Wear OS, Sailfish OS, and the host of server-specific distros. Even today, there are well over 200 active Linux distros, and they're as diverse, interesting, and wonderful as the communities that use them.

Linux Mint became one of the most popular distros by, unbelievably, giving users what they wanted!

Looking forward

But what of the future? Technology predictions are notoriously tricky, but why would we ever let that stop us? Will Tux still be active in 30 years? We'd say that's a safe bet: even if all development stopped now, people would keep on using it for years if not for decades. There are retro computer systems that are still ticking over almost as long later, and the Linux kernel is far more functional than they ever were.

A more likely scenario is Google, as an example, moving to an alternative kernel – Fuschia, say – though this would likely just be for Android and its IoT devices. Yet even if Google moved literally everything it runs to Fuschia, the Linux kernel is used so widely elsewhere that it would just keep on trucking.

As we've seen, the Linux world is larger than just its kernel. An OS is a whole ecosystem of interconnected systems that have to be developed, tested and packaged in an orchestrated manner. Linux was built on GNU tools and its licence; this widened the appeal of Linux and enabled the kernel with suitable distros to be deployed in such vastly differing devices, from the fastest super computer in the world to a lowly $4 Pi.

The Linux kernel isn't tied to the success of any one corporation. Sure, there's the Linux Foundation and Torvalds himself, but succession has already been put into place to keep kernel development going if Torvalds should step down. And while the Linux Foundation isn't necessary, it's certainly handy to orchestrate and handle funding and trademarks.

Put all of that aside, the reason Linux has succeeded is that it's damn good at its job and everyone can contribute. It's the single greatest software development project of modern times, which doesn't mean it's perfect – it's software after all – but it's continually improved and enhanced, it's strong copyleft open source, it fostered a fabulous community and it's given us all endless opportunities. So keep on enjoying it! LXF

WHAT MAKES THE KERNEL TICK?

Linux is just a kernel, but as **Jonni Bidwell** discovers there's an awful lot going on in there...

Explaining what Linux is to senior management types can sometimes be tricky. An operating system, a monolithic kernel written in C, or a way of life are all valid answers. But so too is, to paraphrase Morpheus from *The Matrix*, "no one can tell you what Linux is, you have to experience it for yourself". Fortunately, the fact you're reading this magazine suggests you have some interest in Linux, maybe even some experience, and so we'll treat you to a kernel deep-dive the likes of which would have management quaking in their boots.

The Linux Kernel is at the heart of all your favourite distros. It's the interface between hardware and software, so it contains all the device drivers, networking stacks and weird protocols required to get your system from the BIOS into a usable state (via the bootloader), and hopefully keep it that way. It has direct access to the CPU and memory, so when userspace programs want to access those, they do so via kernel calls. The kernel then dutifully allocates memory, gets the CPU to do the required sums, and gives back the requested result.

Windows and macOS have kernels, too – it's just that those kernels are much more tightly coupled with their respective OSes. Android uses the Linux Kernel, but buried deep beneath layers of Java-flavoured abstractions that make it utterly unlike a conventional Linux distribution. But enough about lesser OSes. We're here to find out what makes the Linux kernel tick, how it has developed and even how to make it tick faster. So without further ado, on with the show!

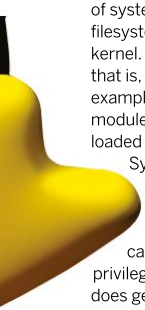
How does the kernel work?

While you're reading this, your computer's kernel is a hive of activity.

When you power on your Linux box the BIOS or UEFI jump into life and do all kinds of checks and initialisations to get it into a state capable of running an OS. Once that's done it hands over to the bootloader (GRUB or systemd-boot, say) whose job is to load the kernel into memory, start an initial RAM disk or filesystem (initrd or initramfs) and then hand over to the init process, known as PID1 since it's the first process to start. And also the process that spawns all other processes.

It's not always strictly necessary to use an initrd, but most distros do this because it simplifies the process of accessing the root filesystem, which might need certain drivers to be loaded before it can be accessed. This so-called early userspace contains everything required to detect hardware, load modules and mount the root filesystem. Oh, and if you've ever used the hibernate function, which dumps the current system state to disk, then it's the initramfs that ensures that image can be read when you power the system back on.

We've already skipped way past the kernel by this point, but that doesn't matter, because from the moment it's loaded the kernel becomes the gatekeeper of system hardware. When your initrd goes looking for a filesystem with a particular UUID, it does so via the kernel. When it wants to query what kind of filesystem that is, it does so via the kernel. The filesystem (for example, ext4 or btrfs) and interface (SCSI or ATA, say) modules are packed into the initramfs so they can be loaded (again by the kernel) and before you know it Systemd (or whatever is your PID1) has taken over your system. Except it hasn't, because everything Systemd and everything that comes after does is done through the kernel. Kernel calls run in a privileged mode, which is even more privileged than the root account if you like, since all root does gets vetted by the kernel too.



From the point of view of a regular user, who deals in terms of command line programs and GUIs, it can be hard to see how the kernel, or even low-level system libraries, fits in. So let's consider what happens when we run a simple `ls` command. The **/usr/bin/ls** binary is called by the shell (for example, *Bash*) and then interfaces with the system C library (Glibc on desktop Linux, uclibc on embedded systems, Bionic on Android). This then issues lots of system calls to the kernel. It may help to consider first the library calls before we talk about the system ones.

The kernel.org team maintain several branches of the kernel, keeping all our devices strong and stable.

First, and this is oversimplifying matters slightly, `opendir` is called on the current directory, which causes a cascade of `readdir` calls that examine all the files and directories there. In order to print the names of these out they need to be read into memory via the `memcpy` function, which requires the length of the filenames to be computed via `strlen`. The length of each filename is used to allocate a buffer (plus one null byte to mark the end of the string). All these names are then thrown back at the shell, which helpfully lists them nicely.

KEEPING YOUR KIT IN CHECK

"From the moment it's loaded the kernel becomes the gatekeeper of system hardware."

(SYS) CALLING ALL UNITS

Armed with some insight into what Glibc function calls are, we can now go one level deeper and begin to get a handle on some kernel calls. Going back to our directory listing example above, the first the kernel sees after the command is issued is an `execve` call. This is how all programs start and along with the filename and any command line arguments, a bunch of extra environment variables are also passed to this call. The `opendir` library call effects a series of `stat` kernel calls, which determine the status of files.

Directories are files too, and the current and parent entries in a directory, denoted by one or two dots respectively, will get `stat`-ed along with any 'real' files. Every system call returns a value (sometimes it's zero), and the value of the `openat` syscall, which opens a directory, is a file descriptor. Armed with this, repeated `getdents` calls gets all the entries for this directory, which can be displayed to a `stdout` device such as your screen or, if you're in a GUI, terminal (emulator) via a `write` call.

You can see system calls in real time with the *strace* utility, or if you're more interested in library calls, there's *ltrace*.

The human effort

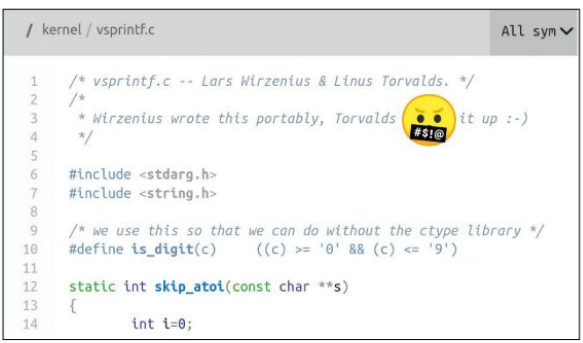
A considerable army is required to maintain the kernel, and that army has got to be managed. Sort of.

The kernel started with just one software engineer, Linus Torvalds, who's still in charge of the whole operation. But as he's said many times before, if he were to be hit by a bus the kernel would still keep being developed. It's worth checking out an analysis of this statement with the fine example of Y2k humour at www.crummy.com/writing/segfault.org/Bus.html ,

Torvalds' day-to-day work is more signing out patches than writing critical functions, and the cadre of subsystem maintainers would be able to manage things if he were to suddenly stop contributing to Linux. So the so-called Bus Factor isn't really a problem for the kernel so much as it is for smaller projects.

The first version of Linux, 0.01, comprised 10,239 lines and 88 files. Today the kernel 5.13 sprawls some 30 million lines and this release alone included contributions from over 2,000 developers (more than 300 of which were first-time contributors). Over the whole of 2020, some 4,000 people contributed. It's hard to get a contributor count across the whole history of Linux, since prior to 2002 there was no versioning system in place. But somewhere around 20,000 seems reasonable. At any rate, it's absurd to think one person could keep track, let alone be in control, of all this development. And indeed Linus is not – he just happens to have the final say in matters of process.

Interestingly, a number of functions remain unchanged since that first release. The file **vsprintf.c** is the only file from v0.01 not to have been entirely created by Torvalds. As he said in the Usenet announcement, it was "written mostly by yours truly except for the vsprintf routine which was co-written with Lars Wirzenius". That function is found unchanged in today's kernel. See for yourself at **https://github.com/torvalds/linux/blob/master/lib/vsprintf.c**.



The elixir.bootlin.com site enables you to pore over kernel code version by version. This classic Linus-ism is still present.

Looking back in time

Lots of functions from the early days are still around, but since others in the same file will change you need to do a bit of work to see what they are. Fortunately people, in particular Dr Daniel German through his cregit work, have tokenised all the many definitions, structures and more, and it turns out that out of the hundreds of millions in today's kernel, just under 3,000 are from that first release. And many more came from not long after. Who says good code doesn't last?

In 1996 Linux introduced the **MAINTAINERS** file, to show who was responsible for what. At first there were only three maintainers: Torvalds, Alan Cox and Jon Naylor. Today, the kernel is divided into several subsystems, including storage, device drivers, USB and Bluetooth, each with their own maintainers (there are about 150 in total). But within each subsystem, another layer of maintainers are responsible for different components. Today's kernel lists over 1,500

» KERNEL TITANS

Besides Linus, there are a few kernel veterans that, having devoted a good chunk of the past three decades to Linux, are still very much active contributors. In charge of the staging tree (mentioned above), as well as several other subsystems (and indeed the stable branch of the kernel) is Greg Kroah-Hartman (gregkh). Storage is the purview of Ted T'so, who's the maintainer for the Ext4 filesystem, as well as the *e2fsprogs* suite of userspace programs for that and its predecessors. Ted works at Google on filesystems and storage, and in 1994 created the **/dev/random** interface.

Alan Cox (of the original **maintainers** file) retired in 2019, but was active until then. The former maintainer of the network stack (which he rewrote in the early days) now runs Etched Pixels, a model railway company. In 2014 he launched the tiny Fuzix OS, which he still develops at **https://github.com/EtchedPixels/FUZIX**. Lars Wirzenius, who we introduced earlier, went on to found the Linux Documentation Project and is now a senior software engineer at the Wikimedia Foundation. He was a Debian maintainer up until 2018, and still contributes to numerous open source efforts.

We met Greg-KH back in 2019. Very friendly, 10/10, WLTM again.

maintainers, so responsibility is shared across many shoulders. Since 2004, the kernel has adopted a standardised Developer Certificate of Origin (DCO) for each commit. This shows who wrote the code and, together with the move to Git in 2005 means that the provenance of contributions can be tracked much more readily. By signing a commit thusly (and it won't be accepted without this attestation) contributors agree to the Linux Foundation's DCO, which can be read at **https://developercertificate.org**.

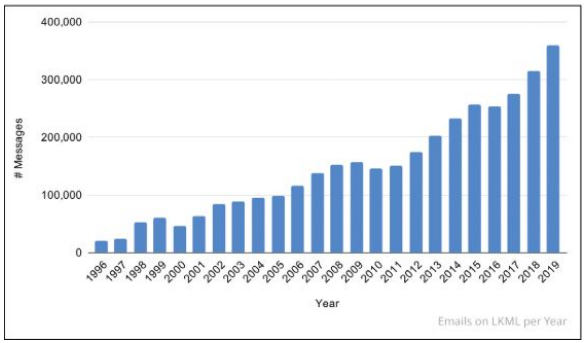
The decision to mandate DCO signoffs stems from the SCO vs IBM, Red Hat, Novell et al (there's a lot of them) legal wrangles that began in 2003, with a baseless accusation that code from System V Unix had been co-opted into Linux. Things took a turn when it was discovered SCO didn't even have rights to most of that System V code, took another turn when Microsoft contributed a cool hundred million dollars into SCO coffers and ended with, ah well technically they haven't ended yet. Even though the SCO Group filed for bankruptcy in 2007 and no longer exists, its remaining intellectual property having been bought by UnixWare and OpenServer Vendor UnXis (now Xinuos) in 2011, after it lost a second appeal against Novell. In April 2021 Xinuos rekindled the 18-year-old claims against IBM, and rails at its acquisition of Red Hat to boot. The protracted zombie case continues.

A great deal of kernel work is now done by robots. They might not be writing patches yet, but they are reporting bugs. Cast your eye over the 5.12 development statistics report at **https://lwn.net/Articles/853039** and you'll see that the top three bug reporters are all bots. And so are positions 4 and 10 by the way. These "fuzzers" come up with fiendish sequences of kernel calls that may or may not be realistic in real life, but nonetheless manage to crash it. Syzbot (the number two reporter) is built around Syzkaller, Google's open source kernel fuzzer. Syzkaller is now used to test Mac, Windows and Fuschia kernels too.

You've got mail... lots of mail

One thing that hasn't changed in a long time is that new kernel contributions are still submitted through the Linux Kernel Mailing List. Email might be a relic for many, but if you still use your inbox and want to see it (very) full, sign up to LKML by sending a subscribe email to **majordomo@vger.kernel.org**. A more sane approach is to just browse the LKML archives at **https://lkml.org**. Of course, maintainers (or anyone) are unlikely to read every single message, so most people will filter messages to the subsystems of their interest. One of the most disheartening things for first-time contributors is having their post ignored, but this generally means no one has seen it rather than it not being worth responding to. To reduce the noise, a couple of subsystems have their own mailing lists (for example, SCSI, Bluetooth and Media). The kernel comes with a Perl script, **/scripts/get_maintainer.pl**, that will tell you which mailing lists and maintainers your patch should be submitted to. It also comes with another script, **check_format.pl**, which will check your patch for various formatting violations and optionally correct them.

Linus wrote Git in 10 days following Linux's departure from the proprietary BitKeeper VCS (version control system) in 2005. Apparently, he named it after himself, showing his appreciation for pejorative terms of the British Isles. After going to all the effort of inventing (the world's most popular) VCS, one would be forgiven for thinking that it seems a little silly to be still handling all these patches over email. Well, it turns out using Git this way, and allowing any old maintainer commit access to the main tree, is a bad idea. Linus does maintain his own Git tree, hosted at GitHub (**https://github.com/torvalds/linux**) as well as on **kernel.org**, and that's what all patches submitted to the mailing list should be based on. If you look in the Pull Requests section of that



With over three hundred thousand messages a year the Linux Kernel Mailing List is nothing if not bustling. Image credit: The Linux Foundation.

> **THE POWER OF TEAMWORK**
> "Today's kernel lists over 1,500 maintainers, so responsibility is shared across many shoulders."

GitHub page, you'll see hundreds of people trying to submit PRs and hundreds of automated responses saying that's not how we do things here.

The Kernelnewbies site is a great resource for any budding kernel developers. It has a guide to submitting your first patch at **https://kernelnewbies.org/FirstKernelPatch**, which covers setting up *vim*, *Mutt*, *esmtp* and Git, and then diving into the testing branch of the staging tree. Things aren't really expected to work here so you'll be sure to find something to fix. Even if you don't have coding experience, if you familiarise yourself with the Kernel coding style at **www.kernel.org/doc/html/v5.13/process/coding-style.html** then you can get started fixing indentation, spellings or other trivial things.

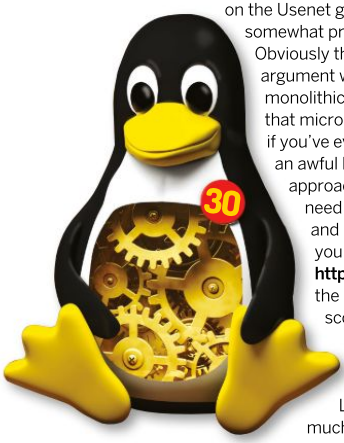
As Linux reaches middle age, there's increasing pressure to find new maintainers. Eventually, barring advances life longevity, the old guard will retire and there'll be full-time shoes to fill. To those brave souls aspiring to such works, we salute you!

Kernel internals

Ever wanted to know how modules and drivers are able to light up your display? Then look no further...

We've looked at how the kernel works, and how people work on the kernel. So how about we finish with some fireside kernel lore. It's always worth remembering that every new project has its detractors (we don't care if you thought the old website was better) and Linux was no different.

The Torvalds-Tanenbaum debates, referred to as a flame war at the time (but both parties stated there was never any personal animosity, despite the colourful language), started when Prof. Andy Tanenbaum posted on the Usenet group **comp.os.minix** in 1992 with the somewhat provocative title "Linux is obsolete". Obviously that title didn't age well, but his argument was essentially that (even back then) monolithic kernels were on their way out and that microkernels were the future. As you'll see if you've ever compiled a custom kernel, there's an awful lot of stuff in there. The microkernel approach would have everything that didn't need to be there – all those filesystems and algebraic operations and what have you – shunted off into userspace. See **https://bit.ly/lxf280-linux-obsolete** for the original debate in all its (really quite scorching) glory and see **http://bit.ly/1cwIB5C** for the battle of the Mr Ts (Round Two).

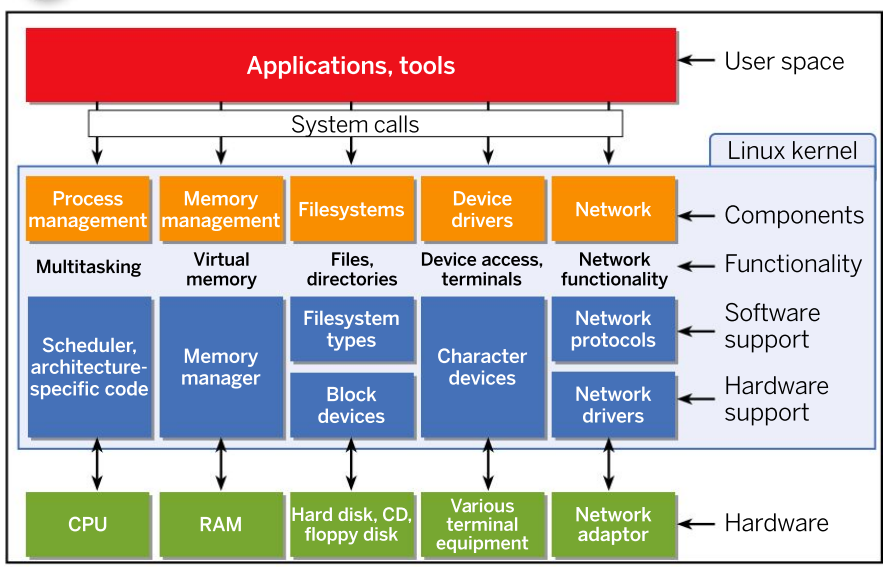


It's an oversimplification to say Linux is truly monolithic. Modular is a much better word. Indeed, if it were properly monolithic every single driver would have to be compiled into the image and then it would probably be so bloated as to not fit in memory. That's why drivers on conventional distros compile drivers as external modules, which are loaded at boot time. You can see which modules are in use on your system (and which modules they depend on) with `lsmod`. If you use the proprietary Nvidia driver (or indeed any out-of tree module or unsigned) then you'll see an appropriately sinister warning that this 'taints' the kernel. Since the kernel team can't be expected to fix bugs that might be caused by code outside their control, any bug reports from tainted kernels are summarily dismissed.

Put away your rose-tinted glasses

Sometimes life is hard for Nvidia users, but the situation is much better than it used to be. Modules need to be compiled against the same kernel version that they're to be used with. So in ye olden days it was necessary to recompile the module before you booted a new kernel. Often the module would no longer compile against this new kernel version. And even more often you would forget to do the compiling in the first place. All culminating in the system booting to a black screen.

Thanks to DKMS (the Dynamic Kernel Module System) this problem has been obviated and new modules are updated automagically. In addition, Nvidia (and AMD's proprietary GPUPRO stack) work much more closely with distributions to avoid any mismatching kernel situations. That open source



Subsystems, everywhere subsystems.

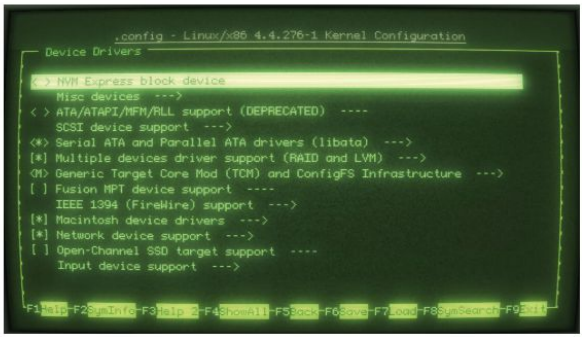
AMDGPU stack may be a boon for gaming on Linux, but it's also by far the largest driver in the kernel, weighing in at 3.3 million lines (10 per cent of the 5.14-rc1 kernel).

This isn't to say it's bloated, sloppy programming, but rather that making graphics drivers is hard. A lot of the bulk comes from automatically generated register header files that come from AMD's own internal specs, so this shouldn't be seen as a burden to reviewers. New header files need to be generated for each new GPU (and they're large), but these are stripped from the kernel at build time, so they equally shouldn't be seen as kernel bloat. There are also a few hundred thousand blank (and comment) lines too, so the amount of 'actual code' as estimated by Phoronix stands at around 420,000 lines. This puts it roughly about the size of the Nouveau and Intel 915 drivers together.

Besides the drivers themselves, a lot of kernel infrastructure is needed to talk to high-grade graphics hardware. In fact, there's a whole framework in X called DRI (Direct Rendering Infrastructure) that enables the drivers to talk directly to said hardware. In the old days of X, it had to do all the rendering since it had exclusive access to the hardware. That doesn't play nice with today's expectations (where applications expect to just throw OpenGL at your card and have spinning platonic solids appear in a jiffy) and the DRI in userspace is supported by the DRM in the kernel. It stands for Direct Rendering Manager, by the way, and has nothing to do with the other, digital freedom-pillaging DRM. DRM gives multiple processes access graphics hardware, whether its for gaming, video compression or crypto mining.

A question of space

Calls to the DRM API are marshalled by libdrm in userspace (and gamers in search of more FPS will often install a newer version of this library). But let's get back to kernel space. One of the crowning achievements of the kernel's yesteryears was promoting video mode setting from user space. Mode setting is the act of setting your display's resolution and refresh rate (back in the day if you did it wrong then the magic smoke might have escaped from your monitor). Allowing user space programs to do this was at best clunky. It resulted in noticeable flickers and occasional distortions, and if for some reason two programs tried to mode-set at once it would be game over. Traditionally, mode setting code lived in X's DDX (Device Dependent X) drivers, but this was suboptimal. For one thing, there was already mode-setting code in the kernel (for example, to set up the classic 80x25 text mode or framebuffer) plus there was this new emerging display protocol called Wayland.



The Curses interface for choosing what goes in the kernel is what Cool Retro Term was made for.

By moving all this code to the kernel (so-called Kernel Mode-Setting, KMS), it meant it could be used by any application and that there was no duplication of mode-setting code. It also meant users weren't at the mercy of janky programming that failed to wake their displays up after suspending and resuming the system. Further work on Atomic operations (see our Wayland of a Time interview with Dan Stone in **LXF243**) not only reduced the flicker between mode changes, but also reduced to zero the chance of a failed resolution change leaving the display in an inconsistent state.

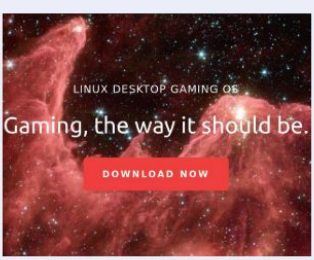
And speaking of inconsistent states, this magazine risks being left in one if we don't get this article off to our very patient printers. Do tell us all of your swashbuckling kernel adventures. LXF

» ACTUAL KERNEL TICKS

Well the headline for this feature turned out to be quite nice, because it turns out the notion of 'ticks' are intrinsic to the kernel. Besides CPU clock speed, which you might think of as the heartbeat of your system (or more accurately one of a few heartbeats), the kernel has its own frequency. Most desktop distributions set this to 250Hz, which gives a reasonable trade-off between responsiveness and not wasting ticks on potentially idle instructions. It's possible to run a so-called tickless system, which doesn't really have no ticks (otherwise nothing would get done), but rather schedules the next tick in response to the next kernel event. "Dynamic ticks" would probably be a better name for this and the idea is similar to dynamic CPU frequencies.

For some tools, for example gaming and professional audio, it's desirable to have a lower-than-average latency. And some distros (including Ubuntu) cater to these situations by providing a low-latency kernel. Amongst other settings, this bumps the tick frequency to 1,000Hz and enables low-priority processes to be pre-empted by higher priority ones others, hopefully keeping the system responsive under load. A related idea is that of a real-time kernel.

But we'd like to squeeze into this box a nod to Drauger OS, an Ubuntu-based distro that aims to provide a premiere gaming experience. It does this with 1,000Hz ticks, the custom Xanmod kernel, bleeding-edge Mesa and Vulkan drivers, and much more. Do check it out at **https://draugeros.org**.



Drauger OS is "absolutely, assuredly still alive," say Drauger OS devs to the surprise of no one.

Dave James is an engineer in the Canonical Foundations team responsible for the Ubuntu on Raspberry Pi images

» GENEROUS COMMUNITY

The Pi Community is one of the Pi's killer features. Aside from the price and technology, the knowledge that if/when your Googling fails you, there's likely to be someone who'll be happy to drop a hint, is invaluable.

When I started working on *picamera* (naively assuming it would wind up a minor Python library for trivial camera usage), it was the community responding enthusiastically to the early releases that spurred me on to take the library further than I'd planned. Later, it was advice from the camera firmware devs (again via the community) that helped me conjure the more advanced features in that library. Most of *picamera* (or my other projects, like *picraft* or *pisense*) wouldn't exist without the aid or support from this community.

Having an active and friendly community has been a major part of Ubuntu's success story too. Years before joining Canonical, I'd switched to Ubuntu as my primary OS, and the community surrounding it was a big part of that decision. Ultimately though, community implies an ecosystem to me: something with mutually beneficial feedback loops (albeit with some outside support, whether that's the Pi Foundation or Canonical nurturing their nascent communities).

One of my great pleasures is fixing something locally in Ubuntu, realising it's generic (something that will affect other distros too) and, without additional bureaucracy, knowing I can feed that back upstream. It makes all Pi user's lives a bit better, and my life a bit easier too.

Pi Foundation boosts computer education

Research project aims to further enhance the stellar educational outcomes from the Pi Foundation.

Our readers will be well aware that computing is an ever-evolving subject. Take the Raspberry Pi; one day there was no Pi, the next it's being used all over the education system in all manner of ways. That in itself is a problem: how best to teach computing?

Out of the problem comes a collaboration between the University of Cambridge and the Raspberry Pi Foundation in the form of the Raspberry Pi Computing Education Research Centre, hosted as part of the University of Cambridge's Department of Computer Science and Technology.

Ensuring UK computer education stays top of the game.

The Pi Foundation states: "Through our research activities we hope to make a contribution to the field of computing education and, as an operating foundation working with tens of thousands of educators and millions of learners every year, we're uniquely well-placed to translate that research into practice."

It'll start by looking at computing curricula and teacher development, non-formal learning and how to remove barriers to computing education. The Pi Foundation says that while being UK based, it's keen to expand research to other countries including the US and India. So if you can help get in contact with **research@raspberrypi.org**. More details at **www.raspberrypi.org/computing-education-research-online-seminars**.

PiStorm Amiga

Kit back from the dead.

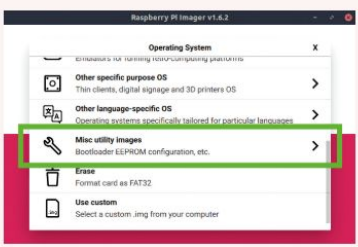
Turn the Pi into a pin-compatible 80MHz 68030 Motorola processor and bring your old Amiga to life! This amazing project uses hybrid software/hardware emulation to offer the Pi as a 680x0 CPU with 128MB of Amiga memory, but also remaps USB devices to the Amiga, hard drive images as Amiga SCSI storage and more on the way. See **https://github.com/captain-amygdala/pistorm**.

An Amiga 500 powered by a Raspberry Pi.
CREDIT: Claude Schwarz

Real USB booting

Firmware update required.

For years the Pi was stuck with running from its SD cards, but finally the Foundation has released a firmware update process that'll enable you to boot a Pi 4 and Pi 400 from a USB device. The *Pi Imager* tool offers a Misc Utilities section with the required firmware. Get a full guide here: **www.tomshardware.com/uk/how-to/boot-raspberry-pi-4-usb**.



Get all the firmware required from the Pi imager.

Raspberry Pi PoE+ HAT

Les Pounder thought the PoE+ HAT was new headwear inspired by the dashing Star Wars hero Poe Dameron. Unfortunately, it is not.

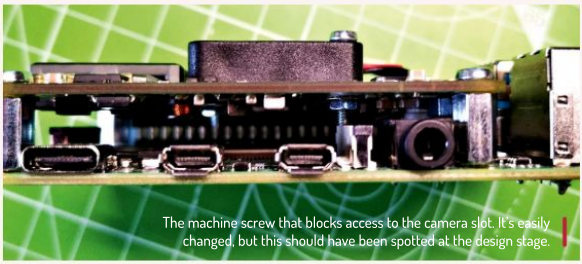
IN BRIEF

The Raspberry Pi PoE+ HAT is an update on the 2018 model, which fixes the annoying fan noise and coil whine but has its own issues that need to be resolved. It has an improved power output that's enough to power even the hungry Raspberry Pi 4.

March 2018 saw the release of the Raspberry Pi 3B+, which featured new PoE header pins. It wasn't until August of that year that we saw the PoE HAT go on general sale. PoE is Power over Ethernet and it provides a one-cable solution for powering your Pi and connecting to a network.

The original PoE HAT worked, but the first batch of units had an annoying "buzzing sound" hardware fault that led to Raspberry Pi offering a replacement scheme. Fast forward to 2021 and Raspberry Pi has released an update to the original model designed with the Raspberry Pi 4 in mind. Yet while it fixes some issues, it creates some new ones, too.

The Raspberry Pi PoE+ HAT is designed for the Raspberry Pi 3B+ and 4, with an emphasis on the Raspberry Pi 4 thanks to an improved power output. The PoE+ HAT works with PoE injectors and PoE-enabled switches that provide between 37 and 57V DC, which is then reduced by the onboard switch mode power supply to the 5V necessary to power a Pi. The amount of available current at 5V has been increased: 4A versus the original model's 2.5A. This gives the PoE+ HAT ample power for a Pi 4, and the possibility of USB-based storage.

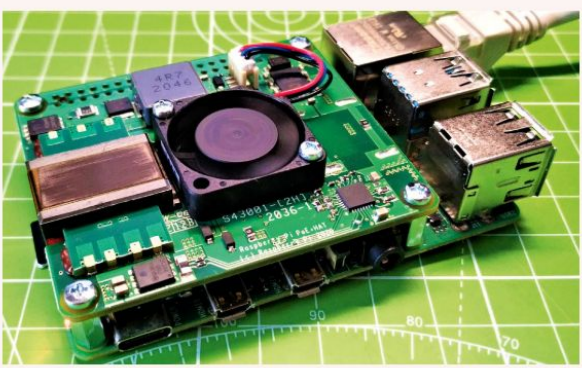


The machine screw that blocks access to the camera slot. It's easily changed, but this should have been spotted at the design stage.

Staying cool under pressure

Dominating the centre of the board is a 25x25mm brushless fan, necessary for keeping our Pi 4 cool. Fan control is automatic, triggered by a sensor reading the CPU temperature, but we do have to add a few lines to our **config.txt** file for the fan to know when to react. We set our config to react when the CPU temperature reached 50°C and the fan kicked into life, keeping our Pi 4 at around 54°C and surprisingly there was no noise from the fan!

As this is Raspberry Pi-designed hardware, it looks right at home on a Raspberry Pi 4 and 3B+.



As you may have already guessed the PoE+ HAT is a HAT and that means it connects to all 40 pins of the GPIO, and an additional four PoE pins only found on the 3B+ and 4. The GPIO is completely covered: there's no access to any of the GPIO pins unless we use an expansion header. It's a hack, but it works. The online instructions tell us to install all four of the M2.5 standoffs to prevent the PoE+ HAT from touching the Pi. But this does introduce an error. The bottom-right machine screw of the CPU fan blocks the Raspberry Pi camera slot. We can replace the screw with a shorter screw, but you would've thought that this would've been picked up in the design/QA process! If you don't use the camera this isn't a problem, but when we consider that the PoE HAT was a popular choice for makers looking to add Pis to remote locations, such as sheds and outdoor projects, this is an irritation at best.

So who is the PoE+ HAT aimed at? The first customers are going to be the networking enthusiasts who want to minimise their wiring and power their Pi-based home server projects from PoE. For those customers, the PoE+ HAT is perfect. It's quiet, easy to install and provides more than enough power for the Raspberry Pi 4. For the makers, the PoE+ HAT GPIO and fan screw issues can be easily mitigated, but these problems should've been solved at the source. For just a little money we can get the screw and GPIO extension ourselves and fix the issue, so don't let it put you off the PoE+ HAT. LXF

VERDICT

DEVELOPER: Raspberry Pi
WEB: www.raspberrypi.org/products/poe-plus-hat
PRICE: £18

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

If you can live with the minor issues and already have a PoE network, then this is a good investment.

» Rating 8/10

SCRATCH

Set up Sense HAT inputs in Scratch 3

Les Pounder shows how to obtain data and control sprites using the Sense HAT add-on board with Scratch 3.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his projects at bigl.es.

YOU NEED

- Raspberry Pi 3B+ or better
- Sense HAT
- Latest Raspbian release
- Code: https://github.com/lesp/LXF280-Sense-Bats/archive/refs/heads/main.zip

This month we'll use the scientific powerhouse Sense HAT board with Scratch 3 to create four quick projects to demonstrate the power of Scratch 3 and the Sense HAT.

The Sense HAT is a remarkable board. Released in 2015, it works with Python, Node-RED and for this tutorial it works with Scratch 3 thanks to a special palette of blocks that enable anyone to use the sensors, display and joystick with very little coding knowledge.

With the power off, connect the Sense HAT to all 40 pins of the GPIO, ensuring that the board fits neatly over the Pi. Use the brass standoff, provided with the Sense HAT to fix the board firmly in place. Connect your peripherals and then power up the Raspberry Pi to the desktop.

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: place a tick in the box and 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, but 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 Sense HAT and a palette of new blocks are added to our code.

The Sense HAT is £25 and it brings an easy-to-use science platform compatible with many programming languages, including Scratch 3.

Project 1: Spin me right round

The Sense HAT has a sensor that can detect our orientation using an accelerometer. This sensor can measure our pitch, yaw and roll, in other words if we pick up the Raspberry Pi and tilt it backwards, forwards, left or right it can measure the orientation. We're going to use the pitch value to spin a sprite. Click the New Sprite icon (blue cat icon in the bottom right) and select the Bat. Then delete the Cat icon using the trashcan icon next to it. From Events, drag `When Green Flag Clicked` into the coding area.

Now from Control drag `forever` to create a never-ending loop. From the Raspberry Pi Sense HAT palette drag `display sprite` and `pitch`. Place `display sprite` inside the loop. From Motion drag `turn clockwise 15 degrees` and place that inside the loop, under `display sprite`. Drag `pitch` and drop it on top of `15`. Click the green flag and look at the Sense HAT on your Pi. There's the sprite! Move it around and watch the sprite rotate.

Project 2: Shaken, not stirred

We can also use the accelerometer to register gestures, for example shaking the Raspberry Pi. From the Raspberry Pi Sense HAT palette drag `when shaken` and start a new section of blocks. From Control drag two `repeat 10` loops, place one under the previous block, then place the second loop inside the other. Change the second loop to `repeat 4`.

Inside the second loop drag from the Raspberry Pi Sense HAT `display Sprite` and then from Looks drag `next costume`. Finally from Control drag `wait 1 seconds` and place it inside the second loop, under `next costume`. So now when we shake the Pi, the Bat will cycle

through all four of its costumes, giving us a basic animation. It does this 10 times. Shake the Pi and watch the bat on the screen and the Sense HAT.

Project 3: Hot stuff

We can use specific actions to trigger our code. Here we'll tilt the Raspberry Pi forward and then change the colour of the bat's nose using the current temperature. The bat will then tell us the temperature.

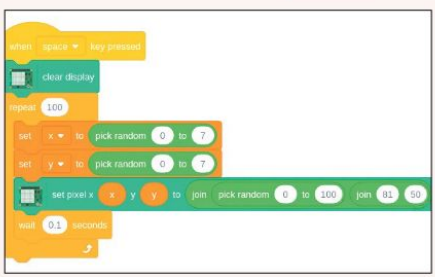
From Raspberry Pi Sense HAT palette drag `when tilted forward` and start a new section of blocks. From Control drag a `repeat 10` loop and connect it to the previous block. From Looks drag `change color effect by 25`. To use the current temperature as a value to change the colour effect we need to drag `temperature` from Raspberry Pi Sense HAT and drop it on top of `25`. Outside of the loop from Looks place `say hello for 2 seconds` and then drag another `temperature` block and drop it on top of `hello`. So now tilt the Raspberry Pi forward and the bat will change colour and tell us the current temperature in Celsius.

Project 4: Disco balls!

The 8x8 (64 pixels) RGB LED matrix of the Sense HAT is awesome, so let's use it to make a random disco dance floor. From Events drag `when space key is pressed` and start a new section of blocks. From Raspberry Pi Sense HAT drag `clear display` and place the block under the previous. From Control drag `repeat 10` and place it under the previous blocks. Change the 10 to 100.

Go to variables and create two new variables for all sprites. The first variable is called x and the second is y. These two variables will store the x- and y-coordinate for a randomly chosen pixel on the matrix. Drag two `set my variable to 0` from Variables and place them inside the `repeat 100` loop. Change the variables from `my variable` to x and y, respectively. From Operators drag two `pick random 1 to 10` and place them over the 0 of each `set x / set y to`. Change the values to read 0 to 7 for both variables.

The next block is a little tricky. It sets the value of a specific pixel using x- and y-coordinates, to a randomly chosen colour. For this we need to drag `set pixel 0,0 to color` from the Raspberry Pi Sense HAT and place it under the previous blocks, inside the `repeat 100` loop.



Project 4 is the most complicated of the four. We use variables to store data, and join multiple items together to control the position, colour, saturation and brightness of the LEDs.

By clicking the bottom left icon we can load extra blocks, including the Raspberry Pi Sense HAT blocks. These blocks simplify working with the Sense HAT board.

Now from Variables drag the `x` and `y` variables and drop them over the `x 0` and `y 0` of the `set pixel` block. To set the random colour grab two `join apple banana` blocks from Operators. Drop the second `join apple banana` block on top of `banana` in the first block. Drag a `pick random 1 to 10` from Operators and drop it over the first `apple` then change the values to 0 to 100. Now change apple and banana to 81 and 50. These set the saturation and brightness of the pixels.

The final block is found in Control and it's `wait 1 seconds`. Place this under the previous block and change the value to 0.1 seconds. Now press the Space key and watch the LED matrix spring into life.

So there we have it – four short projects that show how to use the Sense HAT with Scratch 3. All of these projects can be mashed together to create unique and fun projects. We can control sprites with the Sense HAT, perhaps even our own version of *Flappy Bird*. LXF

QUICK TIP

Stuck and need some help? Download the zip for this project and load Sense-Bat.sb3 via Scratch 3 on your Raspberry Pi. The projects are there, ready to use!

» THIS HAT MAKES NO SENSE!

The Sense HAT was released in 2015, and is a scientific platform created for educational use, oh and in space! There are two Raspberry Pis currently inside the Columbus module of the International Space Station as part of a scheme with the European Space Agency where children can write code and potentially have it run on these high-flying Pis. The competition is called Astro Pi and it has been running since 2015 with code written by school children being run on the two space-hardened Raspberry Pis. Is there a north in space? Using the Sense HAT you can find out, just as one school did. Take a look at the project at **https://astro-pi.org**.

The Sense HAT has sensors to measure temperature, humidity, pressure, magnetic forces, orientation and acceleration. It has a lovely 8x8 grid of multi-coloured LEDs on which we can scroll text and display pixellated images. The joystick is mapped to the keyboard cursor keys and provides us with basic input – handy when you're an astronaut wearing oversized gloves! The board works with every model of Raspberry Pi, from the original B+ to the current Pi 4, and it can be used with languages as diverse as Scratch 3, Node-RED and Python, so every level of user can take their first steps into citizen science. For around £25 this is a great board to get your kids interested in science and space!

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

NEXTCLOUD

Turn your Raspberry Pi into a cloud server

Christian Cawley has had enough of third-party cloud storage. Is a nextCloud-powered cloud server on Raspberry Pi the solution?

OUR EXPERT

Christian Cawley was kidnapped by a giant raspberry and forced to write about small computers. Or he fell in love with the Raspberry Pi and enjoys writing about it. Delete as applicable...

Staying in control of your own data is vital. It goes deeper than personal data, too. All of those files and folders in your personal cloud storage with Google Drive, Dropbox, Box, Microsoft OneDrive and many others are subject to various degrees of – let's say oversight – that you might not be entirely comfortable with.

The solution is to run your own cloud server: a home-based storage system that can be accessed from anywhere. Available 24/7 for syncing data across your devices, mobile and desktop, such a server is best connected directly to your router for reliability. And you can do this with a Raspberry Pi and a suitably sized storage device. While several projects are available for building your own Raspberry Pi cloud server, perhaps the most complete solution is Nextcloud.

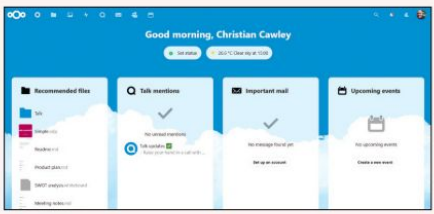
If you're sick of paying for cloud storage or just find the whole idea a bit suspicious, self-hosting your own always-accessible cloud makes sense. With your own hardware and hosted on your own network, the cloud will be completely private. Throw in optional VPN support and even encryption to improve privacy. Self-hosting is also cheaper. For example, a 2TB drive on Amazon is considerably cheaper than the same level of storage with Dropbox or Box.

Rolling your own home-based cloud storage solution will also give you faster speeds. You don't need to rely on internet connections and put up with online traffic and virtual server bottlenecks. The cloud is hosted on your own network, which means that you can sync data based on your router's Ethernet or Wi-Fi speed.

Nextcloud solutions

Several cloud servers are suitable for Raspberry Pi. These include ownCloud, Nextcloud and Seafile. You can even do the whole thing from scratch with the Pi Cloud system, installing various server and VPN software to create a barebones cloud solution on Raspberry Pi OS.

Rather than leave a bare Raspberry Pi hosting Nextcloud on a table top, use a case with built in cooling and space for either a SATA HDD or SSD.



With Nextcloud running on your Raspberry Pi you can enjoy a self-hosted cloud storage, collaboration and productivity tool.

Nextcloud has been chosen for this build. It comes with two installation options, configuration is clear rather than esoteric, and it's open source. When used on your own hardware, Nextcloud is also free to use.

There are three main options for installing Nextcloud on Raspberry Pi: the NextCloudPi image, the Nextcloud Ubuntu appliance for Raspberry Pi, and manually installing on an existing Raspberry Pi OS installation.

While a Raspberry Pi 3 can be used, it's considerably slower than a Raspberry Pi 4. As such, the later Pi is recommended, with at least a 4GB SD card – 128GB cards are around £15. We ran NextCloudPi on a Raspberry Pi 4 with 8GB of RAM and a 32GB SD card. You'll also need a monitor and keyboard attached for the initial setup, although these can be detached, and the server ran as headless after configuration. Ethernet is recommended for speed and reliability.

Get your Pi in the sky

You can take your pick of methods for installing Nextcloud on Raspberry Pi. NextCloudPi is a preconfigured Nextcloud image for Raspberry Pi (and other SBCs) that features everything you need to manage your own cloud data server.

Built with Debian Buster and Nextcloud 19.0.2, NextCloudPi has a setup wizard, web interface, dynamic DNS support, USB automount and everything else you might need to run a cloud server. It also supports Nextcloud mobile apps for mobile data syncing.

NextCloudPi also offers a choice of installation options. You can install it the usual way with an image writing tool such as *Etcher* or *Raspberry Pi Imager*; install with *Etcher* and then move the boot partition to USB; flash direct to USB for use on a Raspberry Pi 4; or use *Berryboot* to flash to SD or a USB device. This guide is more concerned with configuration and use of a Nextcloud server on a Raspberry Pi with the standard SD card installation.

If you're installing using the *Raspberry Pi Imager*, click Choose OS to select the downloaded IMG, Choose Storage to select your SD card, then Write. After flashing, boot the SD card in your Raspberry Pi and login with the Raspberry Pi OS default credentials, these should be changed as soon as possible.

To use wireless networking, enter `sudo raspi-config` and choose 2 Network Options. Select N2 Wireless LAN and choose the correct country, then enter your router's SSID and password. Next, go to 5 Interfacing Options and enable SSH.

Exit *raspi-config*, then make a note of the IP address for wlan0 with `ip addr`. You can then access the Nextcloud server via SSH from another computer using `ssh pi@<YOUR IP ADDRESS>`

The Pi can then be disconnected from the display and monitor at any point now.

Either directly in the terminal or with an open SSH connection, enable the Nextcloud web interface:

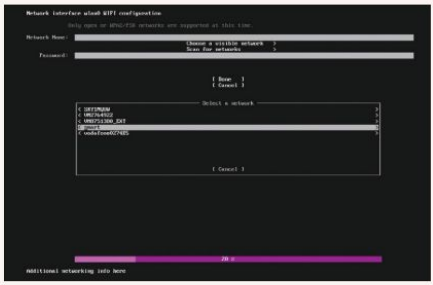
```
sudo ncp-config
```

Select CONFIG then scroll to nc-webui. Erase the no and type yes instead, then Enter. Go Back, then Finish to exit. From your main computer, open **https://<YOUR IP ADDRESS>:4443** in your browser, using the Pi's IP address where specified. Login with the username `ncp` and the password `ownyourbits`. Copy or note the passwords for the web configuration panel and the main web interface, then click Activate.

To use the Nextcloud web interface, open **https://[YOUR IP ADDRESS]**.

Ubuntu Appliance

Using *Ubuntu Appliance* (**https://bit.ly/LXF280-nextcloudappliance**) is a more straightforward approach. Start by writing the downloaded IMG file to your SD card with *Raspberry Pi Imager* (as above) or *Etcher*, then boot the Pi with the Nextcloud image.



Correct wireless configuration enables you to place your Raspberry Pi Nextcloud appliance anywhere in your home or office.

While this starts up, configure your Ubuntu One account (*see walkthrough, page 59, for details*) for SSO (single sign-on) and storing encryption keys for SSH. Before proceeding, create SSH keys:

```
ssh-keygen -t rsa
```

Select where to store the key or tap enter to accept the default location. Input a passphrase when asked.

Retrieve the key using

```
cat ~/.ssh/id_rsa.pub
```

(Windows users can employ *WinSCP* to create an SSH key or open a terminal in Windows Subsystem for Linux and use the above commands.)

Copy the created key string into the SSH Keys page on your Ubuntu One account, then click Import SSH Key to save it.

On the Raspberry Pi, tap Enter when prompted. The next step is to configure networking. With Ethernet connected, simply move on to the next screen by highlighting Done with the arrow keys and tapping Enter. If you're using Wi-Fi, configure this instead.

Next, input your Ubuntu One email (*see the walkthrough on page 59 again*). Shortly after, the account should be verified and the IP address of your Nextcloud server displayed.

Along with the IP address, you should see your Ubuntu One SSO name. This can be used to access the Nextcloud server over SSH, in the style: username@<YOUR IP ADDRESS>.

Full access can be enjoyed via the browser. Using your main computer, open **http://nextcloud.local** in your browser (or the IP address displayed on your Raspberry Pi). Upon first access you'll be prompted to create a

QUICK TIP

Connecting a USB hard disk drive will increase storage for your Raspberry Pi-powered Nextcloud server. A case with space for an HDD is a smart idea, too, such as the DeskPi Pro or Argon ONE (see box, overleaf.)

» SYNCHRONISE YOUR STUFF

Using the desktop and mobile apps to use Nextcloud as a Dropbox replacement is unsurprisingly straightforward. It starts with the creation of a user account (you wouldn't use an admin account!) that you can then login to the app with. With that done, you can begin configuring things to behave like your usual syncing app.

Take image syncing from your phone as an example. Most people using Android with a Google account have a Google Photos auto-sync set up. It's a smart way to stop you from losing important family and personal photos, after all. In the Nextcloud mobile app you'll find an easy menu item to enable this Tap Menu > Settings then find Auto upload. Here, tap the folder on your phone that you want to sync, then allow Nextcloud to run in the background when prompted. Syncing will then commence with your Raspberry Pi Nextcloud.

Manual uploads are also available from the app, along with the creation of folders, new text documents, and even content saved in other apps. For example, you might sync a *KeePass* database with your Nextcloud storage.

On desktop, you can set a sync folder, specify a location on your computer, and decide whether to use virtual files, synchronize everything, or use a selective sync.

QUICK TIP

You can sync calendars to Nextcloud, but don't expect too much from large multi-event calendars spreading several years. These tend not to be imported successfully. It's smarter to start a new calendar in Nextcloud where practical.

server administrator account. This is vital for the administration of the Nextcloud server, enabling you to configure user access.

Terminal install

If you prefer to use your own Raspberry Pi OS for Nextcloud, it can be installed and set up in the terminal. Raspberry Pi OS Lite (**https://bit.ly/lxf280-pioslite**) is recommended for this.

Prerequisites for this approach are the installation of the Apache, PHP and MySQL, along with some PHP modules:

```
sudo apt install apache2 mariadb-server libapache2-mod-php
sudo apt install php-gd php-json php-mysql php-curl php-mbstring php-intl php-imagick php-xml php-zip
```

Next, open the web folder:

```
cd /var/www/html
```

Check the download path (**https://bit.ly/LXF280-nextcloudlatest**) for the version of Nextcloud that you plan to use.

```
sudo wget https://download.nextcloud.com/server/releases/nextcloud-xx.x.x.zip
```

When the download has completed, extract with:

```
sudo extract nextcloud-xx.x.0.zip
```

With this done, configure folder permissions:

```
sudo chmod 750 nextcloud -R
sudo chown www-data:www-data nextcloud -R
```

This ensures that anyone with an account can access your Nextcloud. You can then set up the MySQL database:

```
sudo mysql
```

Create the MySQL user nextcloud, ensuring that a strong password is set.

```
CREATE USER 'nextcloud' IDENTIFIED BY 'password';
```

Next, create the database itself:

```
CREATE DATABASE nextcloud;
```

The nextcloud user should now be given all permissions to the database, again changing 'password' to match the one you set earlier:

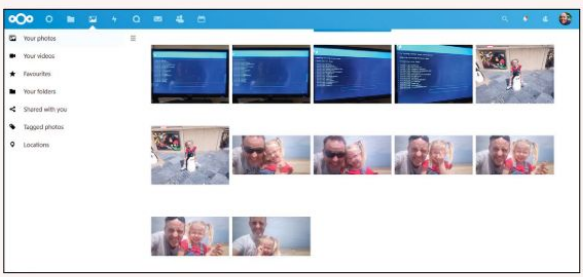
```
GRANT ALL PRIVILEGES ON nextcloud.* TO 'nextcloud'@localhost IDENTIFIED BY 'password';
```

Finish with

```
FLUSH PRIVILEGES;
quit
```

You should now be able to access the Nextcloud server from a desktop browser using the IP address of your Raspberry Pi, **https://<IP_ADDRESS>/nextcloud**.

When prompted, create an administrator account, and input the credentials for MySQL. Click Finish setup and wait while everything is finalised.



Images and videos uploaded or synced from your computer or phone can be viewed and managed in Nextcloud's media player.

Changing clouds

With Nextcloud up and running on the Raspberry Pi, you can connect to it from any browser on your network (or beyond, with a static IP address from your ISP or a tool like *No-IP* or another dynamic DNS service). There's even a mobile app for Android and iOS that can be used to sync files from your phone to the Pi-powered Nextcloud. If you don't use the Play Store, the Android Nextcloud app is also available on F-Droid.

In the browser-based interface you'll find a bunch of menus, presented like a cross between a smart TV and a typical cloud storage UI. From here you can upload files manually from your computer, or via a client tool – available for Linux, macOS and Windows. While command line configuration options are available, everything you need to manage your Nextcloud is available in the browser.

Beyond the default photos, files, contacts, calendar and various other productivity tools, Nextcloud can give you almost anything you need via a collection of programs. Some of these are preinstalled, others can be added in the Apps screen. This covers everything from tools like Collabora Online to new themes, social network integrations, and much more.

To find out more, it's worth taking the time to follow this tutorial with your preferred Nextcloud installation method and explore what's available. Nextcloud has evolved into a vast and impressive cloud productivity environment. Consequently, this guide only scratches the surface of what is possible with Nextcloud on a Raspberry Pi.

» CLOUD IN A CASE

If you're running the Raspberry Pi Nextcloud server with a hard disk drive or SSD attached, it makes sense to have the full setup housed in a case. Various Raspberry Pi cases are now available with space for the PCB and a 2.5-inch drive, some with active cooling solutions, perfect for using the Pi as a data server.

Perhaps most popular is the Argon ONE series of cases, which provide space for a Raspberry Pi 3 or 4 alongside an HDD (**https://bit.ly/LXF280-argon1**). This model also features an internal 30mm fan and aluminium alloy upper case for passive cooling.

Meanwhile, the Argon ONE M.2 Raspberry Pi 4 case enables you to connect superfast M.2 SSD storage to the Raspberry Pi 4, with the same cooling features (**https://bit.ly/LXF280-deskpipro**). This author's personal favourite is the DeskPi Pro (reviewed in **LXF277**. This is equipped with boards for both SATA drives and M.2 (only one at a time, though) and features active and passive cooling.

An alternative, if you have one or can source one, would be one of the WDLabs PiDrive enclosures for Raspberry Pi 3. While missing any sort of cooling, one of these cases makes a tidy alternative.

Whatever Raspberry Pi case solution you opt for, the idea is simple: keep your Pi and HDD in close quarters, reliably powered, and permanently connected to the network.

GENERATE AN SSH KEY FOR YOUR RASPBERRY Pi NEXTCLOUD UBUNTU APPLIANCE

1 Create an Ubuntu One account

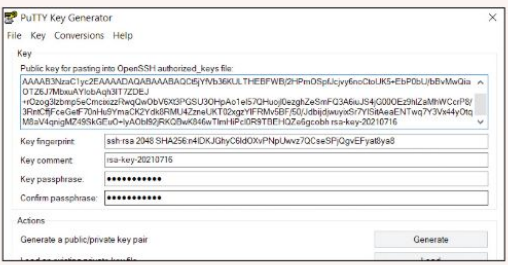
The Raspberry Pi Nextcloud Ubuntu Appliance requires an Ubuntu One account to operate. This ensures a secure, encrypted connection to verify your Ubuntu One login to the appliance. To start, open your Ubuntu One account or create one if you don't already have one (see **https://bit.ly/LXF280-ubuntuone**). Once verified with a valid email address, log in to the account and click the SSH Keys link in the menu.

2 Generate the RSA-SSH key

An SSH key must be created. On your Linux computer, open a terminal and enter:

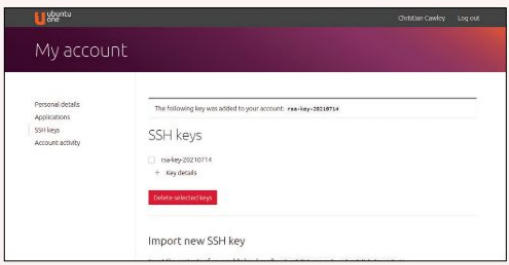
```
ssh-keygen -t rsa
```

This will create a new SSH-RSA key that you can save to your computer. Press Enter to select the default location. You'll need to supply the file path to the desired location if this is different to the default option.



3 Create an RSA-SSH key in Windows

If you're using Windows, key creation requires extra software. Download and install *WinSCP* (**https://bit.ly/LXF280-winscp**) and in the Login screen click Tools>Run PuTTYgen. Here, ensure RSA is selected at the bottom, then hit Generate. Move the mouse when prompted, then input and confirm a passphrase. Use either of the save options to store the key in a dedicated file on your computer.

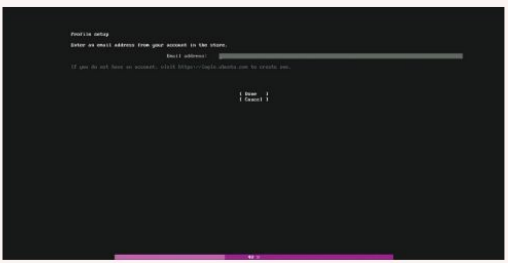


4 Paste the RSA-SSH key into Ubuntu One

Copy the generated SSH key from your terminal or text file. On your Linux setup you can retrieve this with the following command:

```
cat ~/.ssh/id_rsa.pub
```

On your Windows machine, simply open the text file and copy the key string. Then, on the Ubuntu One website, paste the key into the field labelled Public SSH Key and click Import SSH Key.



5 Access your Ubuntu One account in Nextcloud

To conclude secure single sign-on (SSO) authentication of your Raspberry Pi Nextcloud Ubuntu Appliance on first boot, tap Enter when prompted and then log in with your Ubuntu One account credentials. A few moments later, authentication should be complete. If not, tap enter to try again. If the network connection is reliable, it should go through.

6 Connect to the Nextcloud Ubuntu Appliance

With the account authenticated for SSO, you can use the same credentials to access the Nextcloud configuration options over SSH. This affords several options including managing external USB drives. You can use the passphrase you created with the SSH-RSA key to authenticate the SSH connection. Connect using ssh username@<IP_ADDRESS> then enter the passphrase. LXF

MIDNIGHT COMMANDER

Manage files and more

Midnight Commander can do more than just copy, move or delete files. Some would argue that it's as versatile as **Shashank Sharma**.

OUR EXPERT

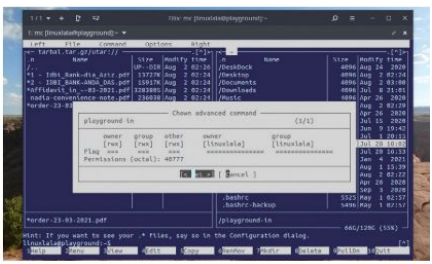
Shashank Sharma is a trial lawyer in Delhi and an avid Arch user. He's always on the hunt for memorable geeky memorabilia.

By standing orders of *Linux Format's* beloved Editor (*your cheque's in the post – Ed*), these two pages are reserved each issue to highlight text-based utilities that generally enliven your CLI experience. As we're celebrating 30 years of the Linux kernel this issue, we've decided to focus on a wonderful command-line project that's been around for nearly as long. *Midnight Commander* is an orthodox file manager that features a split-pane interface and is one of the oldest file managers that's still being actively developed.

Currently under the GPLv3+ license, *Midnight Commander* is part of the GNU project. First released in 1994, *Midnight Commander* quickly became a favourite of most of the Linux crowd. Despite its soaring popularity, with the ever-increasing focus on GUI applications in the late 90s to early 2000s, *Midnight Commander* slowly fell away by the wayside.

This explains why none of the popular desktop distributions offer it as part of the default installation. And yet, you can gauge the continuing popularity of the application from the simple fact that all distributions feature it in their software repositories, making it a breeze to install.

If you're on a Debian/Ubuntu based distribution, you can run the `sudo apt install mc` command to install it.



Apart from running chown and chmod operations with mc, you can use the Advanced Chown feature, which combines both functions.

You can similarly use the `sudo dnf install mc` command to install *Midnight Commander* on RPM-based distributions such as Fedora.

Navigating the MC interface

Once installed, you can run *Midnight Commander* with the `mc` command. *Midnight Commander* defaults to a colour mode, but the default colour-scheme might not appeal to everyone. If you find the use of blue and green unpalatable, you can invoke *Midnight Commander* without colours using the `mc -b` command. Run the `mc --help-colors` command for more information on the supported colour schemes and how to switch between them.

Although *Midnight Commander* is a text utility, it has full mouse support, which means that you can left-click to select different files, or even access the different menus at the top of the interface.

The interface is split into two panes and you can move between the panes by pressing the Tab key. You'll also find a menu bar at the top of the interface. The File and Command menus comprise various entries to help you perform a number of different operations such as copy or rename a file, change its attributes and find or compare files.

Each pane displays the current directory at the top-left. The panes list the directories at the top, followed by files within the current directory and all hidden files are displayed by default. If you wish to sort the files, whether by extension, size, file name, modify time and so on, or switch the display to tree view, for example, then you can do so from under the Left or Right menu, which govern the behaviour of the two panes.

» USING THE SHELL PROMPT

At the bottom of *Midnight Commander's* interface, just above the list of Function keys, you'll find a shell prompt. You can use this to run all manners of commands from within the file manager. For instance, if you don't want to use the arrow keys to select a directory on each of the panes, you can instead select a pane, using the Tab key, and then simply run a `cd` command to switch to the directory of your choice.

By default, the command prompt is active and awaiting user input, so any time you press a key that's not pre-configured as part of a defined shortcut, you're automatically typing on the shell prompt.

The shell prompt can similarly be used to perform other operations. For instance, you can move files, or even extract or uncompress archives without utilising *Midnight Commander's* menus to do so.

When performing some operations, such as compressing files, *Midnight Commander* will briefly drop to the shell, before returning to the TUI. Press the Ctrl+o key to access the output generated by any of the command operations you perform with *Midnight Commander*. After inspecting the output, you can return back to your file manager by again pressing Ctrl+o.

At the bottom of the interface is a strip comprising a series of actions, such as Help, Menu, Copy, Edit, Mkdir and Delete. Each of these actions is preceded by a number from 1-10. For instance, View is preceded by 3, while Deleted is preceded by 8. These number denote the Function keys on the keyboard. So, if you want to quit *Midnight Commander*, you can press F10. To delete the selected file, press F8. You can similarly press F3 to view a selected file.

You can use the scroll-wheel on your mouse or the up/down arrow keys to move though the contents of the current directory. Use left-click to select a file, or press the Insert key, which can also be used to select multiple files. To unselect a file, move to the file using the arrow keys, and press Insert key again to unselect it.

Unlike graphical file manager, *Midnight Commander* lacks the ability to drag files between directories. You must first select the file, and then choose an operation from the File or Command menu, or one of the actions at the bottom of the interface.

You can press the F9 key to select the top menu, and then use the arrow keys to navigate to the menu and its entries or press Alt+ first letter key combination. For instance. To open the Command menu, you can press F9 followed by Alt+C.

File operations

A file manager can be used to perform a variety of functions, such as copying or moving files, changing file permissions, viewing the contents of compressed files, or even compressing or uncompressing archives.

You can perform all these operations, and more, with *Midnight Commander*. It even supports viewing the contents of text and PDF files, and show you the contents of compressed archives without first uncompressing them. Even better, you can also edit text files using the default editor.

Due to space constraints, we can't cover each of these operations in detail, but it's so intuitive that you'll be able to figure it out in almost no time at all.

If you wish to compress a number of files, first select them using the Insert key. Now, press F2 to open the User Menu. You're presented with a number of choices, and for each choice there's a corresponding keyboard shortcut. With time, you'll be able to straight away use these shortcuts without having to read the different functions they perform.

Select the first option, 'Do something on the tagged files', which open the command dialog. You can now enter the command you wish to run to create the compressed archive of the selected file. For instance, run the `tar -czvf tarball.tar.gz %s` command to create a **tar.gz** compressed archive containing all the selected files. The `%s` parameter ensures that all selected files are added to the archive.

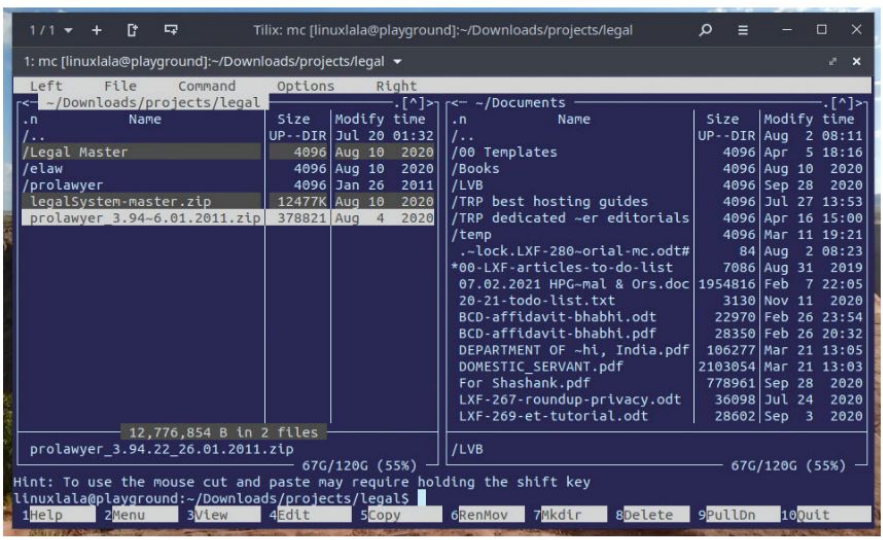
In fact, the main *Midnight Commander* interface also features a shell prompt at the bottom. It's enabled by default, and can be used to perform the same operation without using any menus. You can run any commands on the selected files using this command prompt.

Instead of setting the two panes to display different directories on the local machine, you can also use *Midnight Commander* to connect to a remote machine via FTP. Press F9, select the pane you wish to display the remote machine (Left or Right) and then select FTP.

There's plenty more that you can do with *Midnight Commander*, and while much has been written about this great utility, there just isn't any official documentation. The man pages and the `mc --help` command are the extent of documentation, although the project claims that it's working to improve things on this front. Refer to the man page for quick introduction to keyboard shortcuts that can help you quickly access commonly used operations such as copy and move. LXF

QUICK TIP

When switching between the two panes, you can identify the currently active pane from the directory path at the top-left of the pane, which appears highlighted when the pane is active.



You can use the Options menu at the top to switch from a vertical split to horizontal, and tweak a number of other settings.

SCRIBUS

Credit: www.scribus.net

Sharpen your desktop publishing skills

Aspiring media mogul **Nick Peers** reveals how to design newsletters, flyers and more with Scribus, the powerful open-source DTP tool.

OUR EXPERT

Nick Peers has been dabbling with DTP software since he kick-started his writing career in – gulp! – 1995.

Word processors like *LibreOffice Writer* can do a great job of sprucing up documents, but if you're serious about page design, you can't beat a dedicated desktop publishing tool. *Scribus* is capable of producing anything from brochures and flyers to full-blown newsletters, which you can print or share digitally via PDF as you see fit.

It's packed with powerful tools and options, but in this tutorial we'll introduce you to the fundamentals of using *Scribus*: from changing existing content to designing your own documents from scratch. It can be installed or run several different ways: via AppImage, using its own dedicated PPA, or through Flatpak. While the stable version (1.4.x) is listed as 'recommended', it's been effectively abandoned because the current development version (1.5.7) is on the cusp of replacing it, so we'll be focusing on that version in this tutorial.

The PPA version is best for those running the non-LTS version of Ubuntu, currently 21.04:

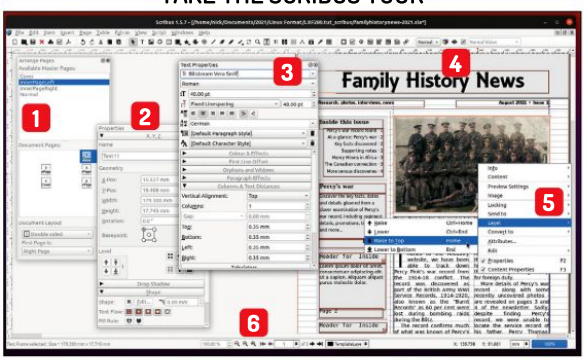
```
$ sudo add-apt-repository ppa:scribus/ppa
$ sudo apt-get update
$ sudo apt install scribus-ng
```

If you're on the LTS release cycle (Ubuntu 20.04) install *Scribus* through Flatpak if you have that installed, or use the portable AppImage if not (**https://sourceforge.net/projects/scribus/files/scribus-devel/1.5.7**). If you're using Flatpak:

```
$ flatpak install flathub net.scribus.Scribus
```

Once installed, *Scribus* can be launched via the Show Applications button. It'll be described as *Scribus* (Beta), but don't worry – it's as close to stable as you can get.

TAKE THE SCRIBUS TOUR



1 Arrange Pages
Use this box to manage the structure of your document – view, add and remove pages as needed.

2 Frame Properties
This box enables you to define the frame's properties. Use the Shape section to flow other objects around it.

3 Item Properties
Press F3 to tweak the properties of a selected frame's contents rather than the frame itself.

4 Page Preview
Click the eye icon shown to show the page without any invisible elements, such as grids and frames.

5 Context-sensitive controls
Right-click an object for more options, including moving it above or below other elements on the page.

6 Navigation and view controls
Use these controls to zoom into and out of the page, as well as move between pages (and layers within a page).

First steps

On first launch, you'll be taken to the New Document window where you have a choice of four options: New Document, New from Template, Open Existing Document, and Open Recent Document. The New Document option enables you to create a single blank page or 'facing pages' (a 'spread', in publishing parlance).

To start using *Scribus*, switch to the 'New From Template' tab where you'll see a selection of templates, split into categories such as books and brochures. Start with a newsletter – pick Newsletter 2 and read through the 'About' description. It's a simple black and white A4 newsletter template where you right-click existing images and text to change it to your chosen content.

Click OK and you'll be warned the file was created in *Scribus 1.3.3* – click OK again. Next, if the Font Substitution box pops up with a warning telling you the document's fonts (Bitstream Vera family) are missing, then cancel opening the document, close *Scribus* and install the missing fonts via the Terminal:

```
$ sudo apt install ttf-bitstream-vera
```

Reopen *Scribus* and create the newsletter template again – it should now work.

Edit texts

The main *Scribus* window will open to show you the first page of your newsletter. It looks rather bare, so let's start by seeing how you edit existing components.

Double-click inside the 'Newsletter Title' text box, where you should now be able to edit the text in the usual manner. Replace this with the title of your newsletter, then select all the text and look to replace the font with something more striking.

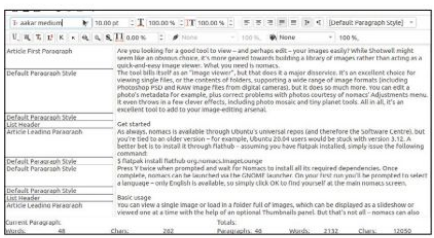
This is done using the Text Properties box, a floating window you can bring up via the Windows menu or by pressing F3. Click the font drop-down at the top to try an alternative font and size – see the box (*below*) for some hints and tips involving font selection.

The Text Properties box has more options too: line spacing will be relevant when formatting multi-column text to ensure it all lines up correctly. Anyone who's used DTP software should recognise most of the options on offer – including the tracking and word spacing controls under Advanced Settings – but for most people the basics should be sufficient.

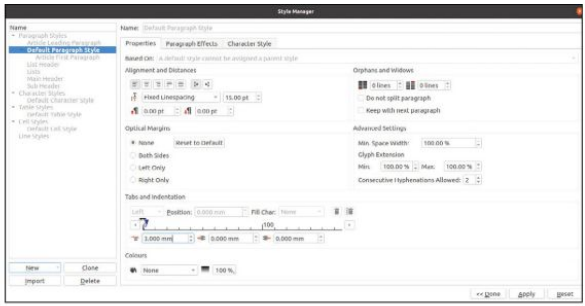
Once you've formatted your newsletter text, you can move on to the rest of the text. When it comes to longer-formatted text, directly editing it in the *Scribus* window can be a little fiddly. Instead, click inside the text box and press Ctrl+T. This opens the Story Editor window with the current text on display inside. It's similar to your word processor, with one key difference – on the left you'll see each paragraph is assigned its own formatting style: Article First Paragraph, Article Following Paragraph, and so on.

Formating text

Consistent formatting is key to creating professional-looking documents, and this can be achieved by applying styles to entire paragraphs. You can change existing styles by right-clicking the current style displayed in the left-hand pane and then clicking the drop-down menu that pops up to select an alternative, but how do you edit these styles to fit your own design?



Use the Story Editor to edit and style your text without having to worry about the layout, too.



Press F4 (or choose Edit>Styles) to open the Style Manager to define both paragraph and character styles. Paragraph styles can be defined from scratch, or you can save time by defining common character styles and applying those to your paragraph styles.

Start by selecting Default Character Style and clicking Edit. This is where you can define the default font that can be used to build your paragraph styles down to the smallest detail. Once done, click Apply – any existing text based on that character style will automatically update, revealing another advantage of styles: you can make wholesale changes without having to reformat everything from scratch.

You can edit existing styles and add new ones of your own. After choosing your default character style, you can then start to build out your paragraph styles. As you'd expect, these add extra elements such as line spacing, margins, tabs and indentation, and text alignment. The Paragraph Effects tab is where you can define bullet and number lists, as well as choose drop caps. The latter is a great tool to draw people's eye to the start of an article.

You'll also see a Character Style tab – you can select your previously defined character style here, then make adjustments (for example, to choose a different style and size for a header or other page element) or – if you're not bothering to use character styles – define this paragraph's character style from scratch.

To speed up the generation of paragraph styles, you can also base new styles on existing ones. They'll then inherit all the characteristics of that previous style, enabling you to concentrate on defining the characteristics unique to that specific style. You could, for example, define a default paragraph style, then use it

Use paragraph and character styles to ensure your text is formatted correctly and consistently.

QUICK TIP

Scribus offers a third type of frame: Render Frames. These make it possible to easily insert specialist typesetting, such as musical notation and scientific formulas, into your document with the aid of third-party programs. Visit https://wiki.scribus.net/canvas/Help:Manual_Renderframes for a guide to using it.

» DESIGN TIPS AND TRICKS

When it comes to designing a document in *Scribus*, one rule of thumb applies: less is more. This means keep things simple for your reader, and prioritising readability and clarity over all else. Start by keeping your page layouts minimal and clean, and ensuring page elements don't crowd each other out. Consider restricting the number of columns on the page to just two or three, and make sure the text flows logically between them.

Readability is key when picking fonts. Again, keep the number to a minimum by using variations from a single font family – bold, black, italics and so on – to style different elements such as headlines or picture captions.

Make sure the font family you choose matches your document's theme – Comic Sans is rarely the best choice for a work-related or professional document, for example. One place where you can look to be a little different is your logo or masthead, although again this should reflect your document's overall theme.

A key choice you'll need to make is deciding between serif (traditional, established) and sans serif (modern, clean) fonts. Which one you choose depends on your content, but if you're still struggling to choose, visit **https://bit.ly/lxf280-font-advice** to read some expert advice on the subject.

QUICK TIP

Save time recreating frequently used elements: group, then choose Windows> Scrapbook to open the Scrapbook window. Drag your element on to here to store it securely. Drag the element out of the window to create a copy to reuse.

to generate additional paragraph styles such as the first article in a paragraph, or a crosshead – a sub-heading within the text itself. Anything specifically defined within the new style overrides the style it's based on.

Working with images

Our newsletter template has space for a single image. To fill this with a new image, right-click it and choose Content>Get image (or press Ctrl+I). The image will be placed full-size within the picture frame, so will require adjusting. Right-click the image and expand the Image sub-menu where you'll be able to adjust the frame to the image or – more likely – resize the image to fit the frame (choose Adjust Image to Frame). This will resize the image to fit the frame, while respecting its aspect ratio, so there will likely be space below or to the side of the image depending on its aspect ratio.

Other options you'll see include Image Effects (add up to 10 different effects, then tweak each one using the preview to see the effect they'll have), Edit Image (opens the image in the image editor you specify under File> Preferences>External Tools, which is set to *GIMP* by default), and Embed Image. Selecting this will convert the image from a link to an actual object within the file. You'd do this if you wanted to share the document with someone else and not have to worry about including all the images as separate files.

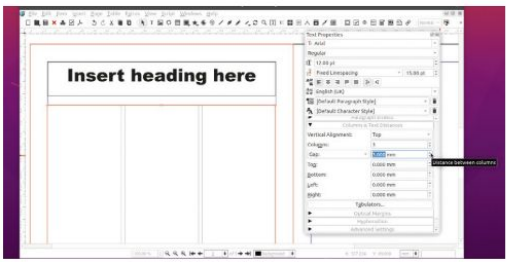
Understanding frames

So far we've respected the newsletter's layout and focused on populating it with content. You'll notice that text and images reside in frames, and that these can be moved, resized, rotated and more – press F2 to open the Frame Properties window to access these options.

With images, we've already seen how we can resize images to fit inside their frames, leaving some white space beneath or to the side. Complete the job by right-clicking the image again, but this time choosing Image> Adjust Frame to Image. You'll see the frame now fit around the image, leaving white space beneath it.

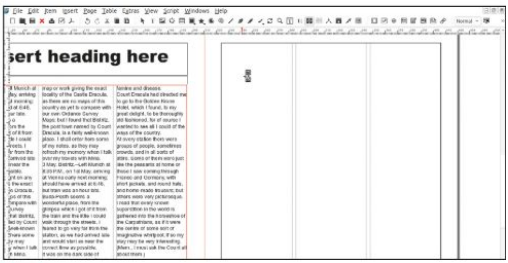
The next step is to move the caption frame so it sits underneath the image again. By default, the template has locked all its frames so they can't be accidentally moved. To fix this, right-click the frame and choose

WORKING WITH FRAMES



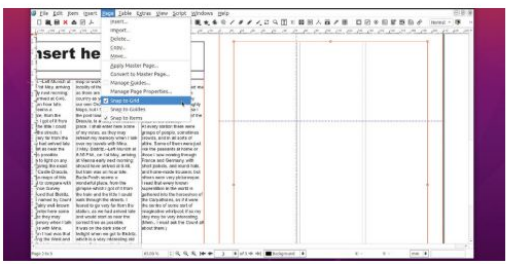
1 Set up a text frame

After opening a blank document, click the text frame button, then click and drag on the page where you'd like the frame to appear. This can then be resized and moved as required. Press F3 to bring up the Text Properties box to set other attributes including text columns (under Columns & Text Distances). Be sure to set a gap between columns as shown.



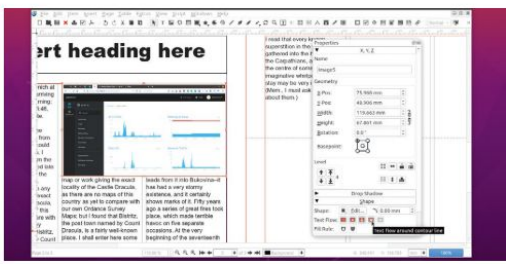
2 Link text frames

If you want to flow text between different frames and/or pages, you need to create links between them. Select your first frame and press N – you'll see the cursor change. Simply place this over your second text frame and click to link them. Repeat the process for any other frames. You can unlink frames by pressing the U key instead.



3 Make use of guides

To ensure frames line up correctly, create guides to help – click and drag from the vertical or horizontal rulers to the left and top of the page respectively to create them. As you click and drag, you'll see their exact X or Y position is shown to help you position them precisely. Choose Page>Snap to Guides to help align frames more easily to guides.



4 Layer frames

Frames can overlap or be placed one on top of another thanks to *Scribus*'s support for layers. Right-click a layer and choose Level to move it up, down or to the top or bottom of the current pile. This affects how content in other frames is displayed – for example, to flow text around an image, place the image at the top, press F2 and set the flow using the Shape tab.

Locking>Is Locked. You'll see the frame change to reveal drag handles – click and drag it to a new location, or use the arrow keys to nudge it around the screen.

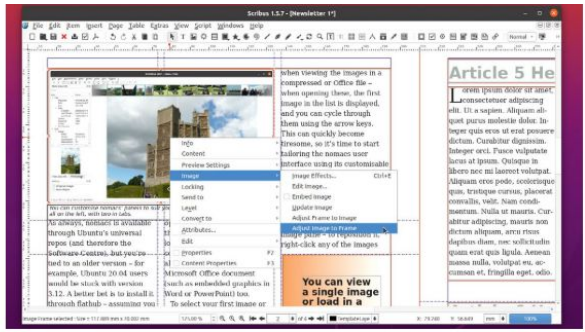
Now's the perfect time to discover how you can group frames together, so they act as a single object. Doing so would enable you to move the image and its caption as one: just hold the Shift key as you click each frame you'd like to group and you'll see the selection box encompasses all selected frames. You could drag this to a new location, or choose Item>Grouping>Group to convert them into a single object (you can undo this from the same sub-menu, choosing Ungroup instead).

Next, drag both the headline and the body text frames beneath so they sit underneath your image (select both, then unlock them and finally move them into position). You now have space left over beneath the text, so select the body text frame to click and drag the bottom of the frame down to fill the space. You'll end up with room with more text to fill too.

Add more pages

Once you've filled your first page, move on to the next. Each template can support multiple page layouts, and our newsletter is no exception. Choose Page>Insert. You can insert one or more pages after the current page and choose different master pages for both left and right-facing pages. For the purposes of producing a four-page newsletter, insert 3 pages, choosing InnerPageLeft and InnerPageRight as your page templates. Click OK.

Three new pages will be created using the layouts you specified, ready for you to populate them with content. This is all very straightforward, but only if you're happy to work within the confines of the newsletter template itself. In reality, you'll want to make changes to this layout following the frame tweaks we've discussed, or look to make bigger changes by designing pages yourself – the walkthrough (*opposite*) reveals how to create text and image frames from scratch.



Scribus makes it easy to fit imported images into an image frame. It's also possible to embed the image directly into the document.

Once you've started to master page design, you'll want to incorporate those layouts in your own custom templates. The box (*below*) reveals how to build a template using a series of master pages, which are used to pre-populate pages with non-editable content.

Share your creation

Your work is complete, and you're ready to print a physical copy or export it as a PDF to share digitally. The best way to do this is via File>Print Preview or File>Output Preview (for PDFs). A Preflight Verifier window will pop up listing all known errors – including those frames where there's too much text to fit. Click an item to jump to it in the document, see the problem and correct it if necessary, then click Check again.

Alternatively, click Ignore Errors to see how the document will look – you'll see various additional options depending on whether you're printing or exporting to PDF – and finally click Print or Export... to complete the job. LXF

QUICK TIP

Want to add page numbers to your document? Create a text frame in the usual way, then choose Insert> Character>Page Number or press Ctrl+Shift +Alt+P. The inserted page number will update automatically if the page is moved around.

» BUILD A TEMPLATE

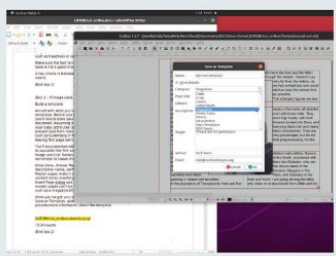
Armed with what you've learned in this tutorial, you can now create your own templates. Before you begin, choose File>Preferences>Paths to specify where you'd like to store your templates. Once done, click OK and create a new blank document.

Assuming it's a multi-page document, select Facing Pages, choose your basic attributes (size and orientation) and use the Margin Guides tab to prevent text from running to the edge of the page (use one of the Preset Layouts such as Gutenberg or Magazine if you're not sure). Choose 3 or 4 for your pages, leaving first page set to Right Page to accommodate a cover. Click OK.

You'll be presented with a blank canvas. Follow the walkthrough (*facing page*) to discover how to populate this first cover page with your own design using a combination of image and text frames. You can then repeat the process for the inside pages – remember to tweak the design for left and right pages.

Once done, choose Page>Convert to Master Page. Give it a suitably descriptive name, verify the right type (left or right) is selected and click OK. Master pages make it straightforward to pre-populate pages with previously created content when inserting new pages. Just select the chosen master page from the Insert Page dialog and *Scribus* will do the rest. Note that content added to master pages can't be edited or removed, so use them for consistent elements, such as a magazine title or page numbering.

Once you've got your pages and master pages in place, choose File>Save as Template, give it a suitable name and optionally tick More Details to provide more information about the template.



You can record details about your new template – perfect if you plan to share it with others.

BACK ISSUES » MISSED ONE?

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).

ISSUE 275
May 2021

Product code:
LXFDB0275

In the magazine
Get more from your Raspberry Pi with our great selection of projects. A parents' guide to coding, an introduction to 3D tool *Blender* and a tour of video conferencing tool *Jitsi Meet*. Plus code a first-person shooter, audio software reviewed and photo effects.

DVD highlights
Mageia 8 (64-bit) and AntiX 19.3 (32-bit).

ISSUE 274
April 2021

Product code:
LXFDB0274

In the magazine
Ramp up the security of your Linux system and turn it into a fortress. Have fun with a Raspberry Pi Pico, explore the best window managers, code a *Space Invaders*-style game in Python, emulate 486 PCs, set up a back-up system and more!

DVD highlights
Linux Mint 20.1 (64-bit) and Sparky 5.14 (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

QEMU

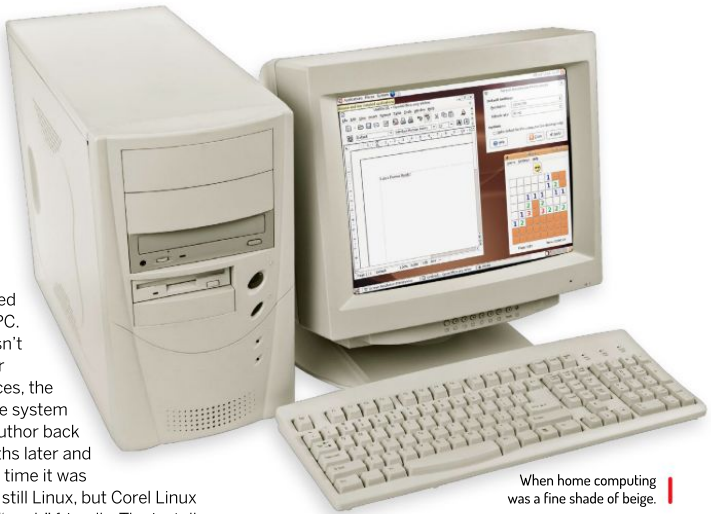
Credit: www.qemu.org

How to run classic distros with QEMU

Where we're going we don't need roads, we need QEMU and a bunch of retro Linux ISOs, reveals **Les Pounder**.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his projects at bigl.es.



When home computing was a fine shade of beige.

Back in the late 1990s this author chose their PC magazines by what was on the cover disc, something that had started back in their Amiga days. One month they chose a PC magazine that had something called Red Hat Linux on the second disc which was promptly installed on an AMD K6-2 333MHz PC.

It's fair to say that it wasn't love at first sight. The sheer volume of application choices, the differing commands and file system was enough to scare this author back to Windows 98. A few months later and another coverdisc, and this time it was something different. It was still Linux, but Corel Linux 1.0 which was a little more "noob" friendly. The install went well and this was this author's distro for quite some time.

In the 1990s, Linux was still in its infancy and the jump from Windows to Linux seemed massive and exciting. How can we experience these days again? In the absence of a giant Tux-shaped time machine, we can use virtual machines to emulate hardware of the era and install Linux on a virtual PC.

We've chosen three Linux distros from the past three decades of Linux and using virtual machines we shall install and use each distro. Our choices span the 1990s, 2000s and 2010s and show the similarities, and differences between Debian, Ubuntu and Linux Mint of these eras.

QUICK TIP

Want to share a USB device with your virtual machine? You can easily do that via the Virtual Machine> Redirect USB Device menu. Handy for using USB flash drives with your virtual machine.

How To Install QEMU

QEMU is a generic and open source machine emulator and virtualiser. In other words it can emulate the hardware of many different CPUs and machines, and create virtual machines. To install *QEMU* we need to open a terminal and use the package manager (in our case Ubuntu 21.04 and the *apt* tool) to install *QEMU*, a Kernel Virtual Machine (KVM) and its dependencies.

Why are we installing a KVM? It's because *QEMU* is an emulator, and so by using a KVM and having a CPU that supports Intel VT-x or AMD V we can speed up the virtualisation to near-native speeds. Most modern processors have some form of support for virtualisation – our machine is an i7 3770 from 2012 and it supports VT-X.

To check if your Intel or AMD CPU supports these features, open a terminal and type the following. For Intel VT-X, use

```
grep --color vmx /proc/cpuinfo
```

and for AMD V, type

```
grep --color svm /proc/cpuinfo
```

Now install the *QEMU* application with KVM

```
$ sudo apt update
$ sudo apt install qemu-kvm libvirt-daemon-system
libvirt-clients bridge-utils
```

To make managing our systems a little easier, we'll install a GUI manager for our virtual machines.

```
$ sudo apt install virt-manager
```

We now need to reboot our machines for the changes to take effect. If we skip this step then the GUI manager will be unable to connect to *QEMU*.

Open the *Virtual Machine Manager* and take a look around. In the top left we have an icon – a monitor with a yellow star – to quickly create a new virtual machine. We shall be using this icon quite a lot in the tutorial.

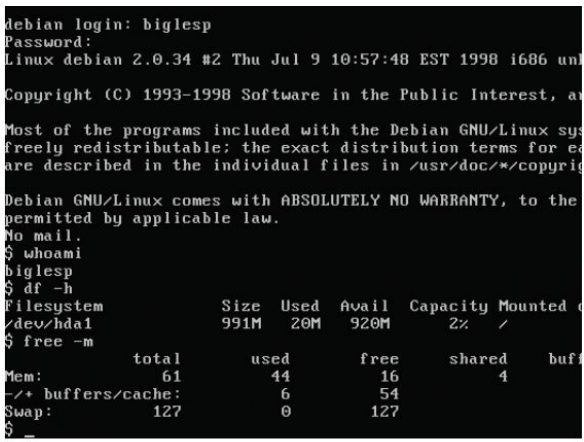
Virtual machines are listed in the main section of the window and we can select and run machines from here. Selecting a machine and then clicking the Play button will start the machine in the background. Clicking Open will open a new viewer window – our interface with the virtual machine. Launching a machine will launch a player window, where we interact with the virtual machine. In this player window we can administer our virtual machine: redirect host USB devices to the virtual machine, configure the system with more RAM, better CPU, insert CD/DVDs and configure our networking.

The 1990s – Debian Linux

Linux in the 1990s was the new frontier. New distros appeared that took advantage of 486- and Pentium-era machines, while downloading the ISO images took hours, if not days over a dial-up connection. Luckily for most of us in 2021 we have much faster broadband-enabled internet.

Debian Linux was created by Ian Murdock in 1993 and the first version (0.01) was released that year. But it wasn't until 1996 that we saw the first stable release (1.1). Today, Debian is the backbone for many popular distros including Canonical's Ubuntu and the Raspberry Pi Foundation's Raspberry Pi OS. It's fair to say that without the existence of Debian the Linux landscape would be totally different.

When creating a new VM we can insert an ISO image or CD/DVD ROM. We can also use a qcow2 or img disk image for a ready-made system.



From these humble beginnings, Debian rose to become one of the most important distros in the history of Linux.

We could download and install our own version of Debian Linux, or we can download a hard drive image that we can use with *QEMU*. The latter is the most simple, being ready to go with little configuration. That said, the Debian installation process is sublime, more so in recent years. The installation process takes us by the hand through partitioning, user setup and then we can configure the system for our intended use.

We downloaded the Debian 2.0 image from **https://github.com/palmercluff/qemu-images** and then extracted the *qcow2* file from the archive.

Open *Virtual Machine Manager* and click File>New Virtual Machine or click the icon of a monitor with a star. In the first screen Create a New Virtual Machine, select Import Existing Hard Disk Image and click Forward. In the next screen navigate to the location of the extracted *qcow2* file by clicking Browse, and then Browse Local. Once you've selected the volume select Choose Volume and then type in "Generic OS" under Choose the operating system. Click Forward. Set the RAM to 64MB and the number of CPUs to one. This is more than enough for Debian 2.0. Click Forward and in the next screen name the machine "Debian2" and click Finish to set the configuration and start the machine.

On first boot we go through a typical Debian

QUICK TIP

Need to quickly use a live CD or DVD image? With QEMU we can quickly start the ISO, or a real disc with 1GB of RAM using qemu-system-x86_64 -boot d -cdrom image.iso -m 1024.

» OLD PC HARDWARE

Emulating old PC hardware with *QEMU* is cheap – it only costs us a little hard disk space. But what if we want to use real hardware? This is where things become expensive.

Looking online we can see incomplete 486-era machines trading hands for hundreds of pounds. So that 486 DX2 66MHz you fancy is now just out of reach. Old PC hardware is expensive to ship and unless it's in the hands of a collector, it often needs lots of work to make them ready for use.

But say you do buy an old PC – what do you need to do? First, make sure you know your stuff before starting the job. Take care with electrics and chemicals for restoring a machine. If in doubt, seek the knowledge of an expert. Older machines used AT power supplies: it's possible they're broken, so make sure you have a known working supply to test.

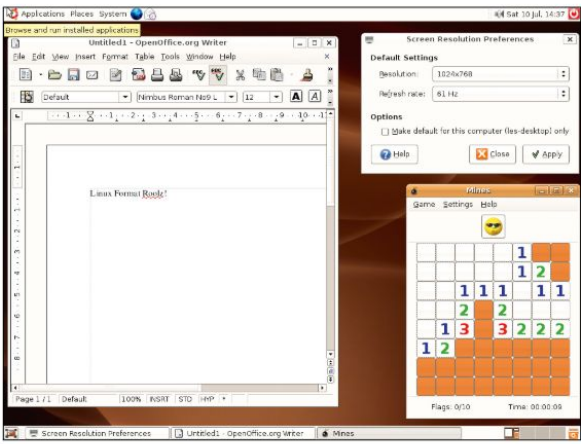
Capacitors may need replacing, and there may be leakage damage to the motherboard, so read up on how to fix this and test on a non-visible area before moving on. The old beige cases may need some retrobright which requires UV light and hydrogen peroxide in order to bleach the plastic to the desired colour. Older hard drives are also prone to dying, so perhaps consider an IDE to SATA or IDE to compact flash/SD card solution.

configuration process. And yes it does say LILO: this author remembers having to learn how to configure that to get their system to boot.

The Debian configuration process starts by setting a root password, then we create a new user account for general access. When prompted to set up shadow passwords, you can select yes or no – it makes no difference. Do remove PCMCIA drivers if prompted. Next we're asked if we'd like to set up PPP, used for connecting to the internet via a modem. Answer N.

The next screen asks if we'd like to run the *dselect* program, used to automate installation. Select No and press Enter. If the *dselect* screen appears anyway, press Q and Enter to drop into a Debian login. Enter your username and password and start using a few commands to mess around with the system. Note that our user has no access to the sudo group; rather, we need to switch user to root in order to make any system-wide changes.

Debian was a refreshing distro at this time. It was easy to install and a joy to use. But our virtual time machine is ready to make a leap forward, to a time when the GUI installer was a big deal to sell the distro.



Ubuntu 6.06 "Dapper Drake" was the first LTS from Canonical. It changed how Ubuntu was perceived and it set the standard for many future releases.

The 2000s – Ubuntu 6.06 Dapper Drake

The first LTS (Long Term Support) release of Ubuntu was a big deal in 2006. It saw Canonical take a stance as a serious player in Linux, and saw many adopting the Debian-based distro as their main OS. In 2006 this author had just been playing with 5.10 for a few months and was ready to make the switch from SUSE. Dapper Drake came along, and despite being a shade of brown they adopted it for their daily workflow. Let's take a look at it all over again. The Ubuntu installation process in 6.06 is barebones and really effective. Fifteen years after it was released it still feels fresh and new.

Download the 32-bit ISO from **http://old-releases.ubuntu.com/releases/6.06.0** before opening *Virtual Machine Manager* and creating a new machine.

In the first screen Create a New Virtual Machine, select Local Install Media and click Forward. In the next screen navigate to the location of the downloaded ISO image by clicking Browse, and then Browse Local. Once you've selected the volume the operating system name should auto-populate to Ubuntu 6.06. Click Forward. Set the RAM to 1024MB and the number of CPUs to two. This is more than enough for Ubuntu 6.06. Click Forward and in the next screen click Finish to set the configuration and start the machine from the Live CD. Now we follow a typical Ubuntu install, with a snazzy GUI installer.

Ubuntu 6.06 should run fine on the default settings, but should we need to tweak them we can break out of the virtual machine by pressing Ctrl+Alt+G and click the blue information icon in the viewer window to open a configuration menu. We can change the RAM, CPU type and insert virtual/real devices into the virtual machine. Note that every change must be applied before moving to the next item in the configuration. Changes to the configuration take effect the next time the virtual machine is power cycled.

Installing software and getting online is tricky. Most of the software repositories are now unavailable, so we need to install programs by compiling the code. Getting online is possible, but this version of *Firefox* is no longer supported and so we can't access any HTTPS sites. But we can still relive the glory years of Ubuntu's rise to dominance, and that lovely shade of brown.

The 2010s – the rise of Linux Mint

Ubuntu had been the darling of the Linux distros for some time, but in the 2010s some grew upset with the introduction of Unity, and an aubergine-based colour scheme. Linux Mint was seen as an alternative. It had a similar back-end to Ubuntu, but a more traditional user interface. Linux Mint also offered pre-installed codecs for popular music and video formats. So via the super simple installer we can quickly get up and running. We downloaded a suitable ISO from **https://mirrors.evowise.com/linuxmint/stable/17.3**.

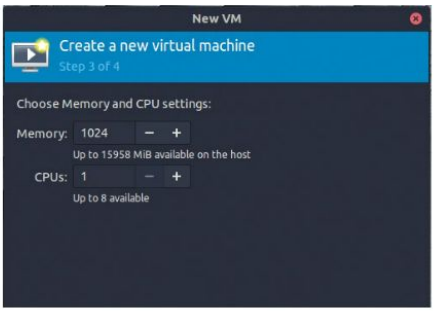
Setting up *QEMU* for Linux Mint is the easiest of them all. In the first screen Create a New Virtual

» VIRTUALBOX

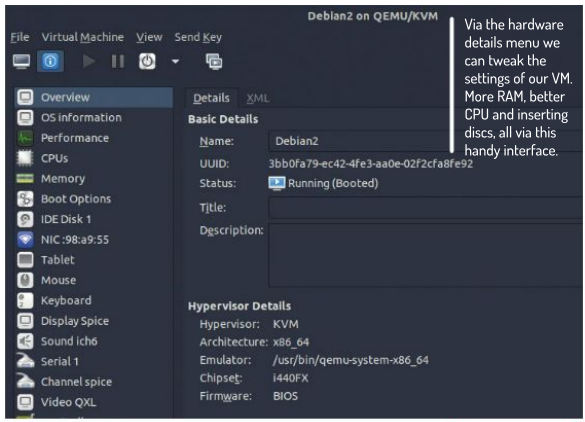
QEMU isn't the only way to create virtual machines. *VirtualBox* from Oracle is a tried and trusted means to create virtual machines, but it can't emulate older hardware as effectively as *QEMU*. Installing *VirtualBox* is a simple task of downloading the latest version from **www.virtualbox.org/wiki/Downloads** and then following the installation guidance. We've successfully virtualised many Linux distros, including Linux Mint and Fedora with *VirtualBox*, and we have also virtualised Windows installs for projects.

VirtualBox has a straightforward process in which we create new machines. During the process we can create virtual hard disks that are either a fixed size, for example an 8GB image file, or we can dynamically create a file hard disk file that will grow to the maximum that the host OS expects. Once a virtual machine has been created we're free to tweak the RAM, number of CPU cores, VT-X or AMD-V acceleration and add ISO images for boot. We can also create shared folders with the host system, enabling easier file transfer. Networking is also handled for us, with no configuration necessary.

VirtualBox is proprietary software and as such it may not be for you. If not, then as we have shown in this feature, *QEMU* is more than up to your virtualisation needs.



When creating a new virtual machine we can set the RAM and CPU. If the OS is detected during the process, it'll suggest the



Via the hardware details menu we can tweak the settings of our VM. More RAM, better CPU and inserting discs, all via this handy interface.

Machine, select Local Install Media and click Forward. In the next screen navigate to the location of the downloaded ISO image by clicking Browse, and then Browse Local. Once you've selected the volume, set the operating system name to Generic Linux 2016 and click Forward. The RAM and CPU will be set to 2GB and two cores – this is based on our choice of "Generic Linux 2016" in the previous screen. This configuration is more than enough for Linux Mint 17. Click Forward and in the next screen click Finish to set the configuration and start the machine from the Live CD.

Now we follow a typical Linux Mint install, but you may hit a snag. In our install the live CD refused to start the X server. This required us to click the blue information icon in the viewer window, navigate to Video/QXL and change the video option to VGA. Apply and then power cycle the virtual machine. Everything went smoothly after this and we were even able to install software and browse the web via *Firefox*.

Running QEMU from the terminal

So far we've created hard drive images and installed distros using the *Virtual Machine Manager*. We can do the same thing using the terminal and *Crunchbang++* (**https://crunchbangplusplus.org**), a modern version of *Crunchbang Linux*.

First we need to create a virtual hard drive. In this case it'll be a 10GB .img image called **cbpp.img** and it'll be stored in the current directory:

```
qemu-img create -f qcow2 cbpp.img 10G
```

We have a blank image ready for our install, so now we need to configure our machine. This is a long command, so we'll go through each option.

The first command invokes *QEMU* with 64-bit support, then we use the `-cdrom` option to tell *QEMU* where to find our *Crunchbang++* ISO. Using `-hda` we tell *QEMU* that we wish to use the **cbpp.img** virtual drive, which is in the current directory. We then set the RAM to 2GB using `-m 2048`, before setting up networking, enabling the virtual machine to access the Internet.

```
qemu-system-x86_64 -cdrom /path/to/iso -hda ./cbpp.
img -m 2048 -netdev user,id=mynet0,hostfwd=t
cp::8080-:80 -device e1000,netdev=mynet0
```

Pressing Enter will run this command and start the boot process. For a first boot we recommend running a live session to test that everything works. After that you can reboot the virtual machine and install to the virtual hard disk.

Linux today: The Undiscovered Country

In 2021 we're truly spoilt for choice. New Linux distros are coming out at an accelerated pace, including specialist distros for media production, networking and scientific projects. Older hardware is supported by lightweight distros such as MXLinux, which prolongs the life of older equipment. Power users aren't forgotten: we have bespoke gaming distros that use *Wine*, *Proton* or *DOSBox* to run Windows games with the latest RTX and RX GPUs. By looking to the Linux distros of the past we can see how our much-loved distros have grown. From the humble text-based installer of Debian to the slick guided installation process of modern distros, our Linux experience is very different in 2021 – but we can still see the rock-solid foundation under the hood. LXF

QUICK TIP

By default, Virtual Machine Manager will store all of the hard disk images in /var/lib/libvirt/images, but we can also specify our own destination when creating an image. Using the terminal it'll default to the current directory from which command is invoked.

Linux Mint 17 is a great balance of form and function. It looks great, and it came with everything we needed to enjoy our computer.

NEWSBOAT

Credit: https://newsboat.org

Read RSS feeds direct from the terminal

Focus on what's important in the world and cut out all those cat meme distractions with **Neil Mohr** and a terminal-based text-only news reader.

OUR EXPERT

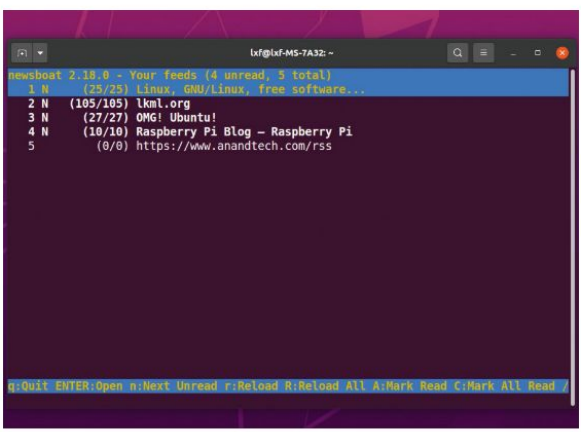
Neil Mohr doesn't get let out much to write articles, so when he does he takes the opportunity to shout for help! But deep down, he probably likes being locked up all day in the LXF dungeon!

What sort of world have we created? An open internet was supposed to be a bastion of human knowledge, where we could all share our thoughts, contribute to the advancement of all human knowledge and have an occasional game of *Quake*. What did we get instead? Facebook and a web of endless adverts and faux top-ten buying guides.

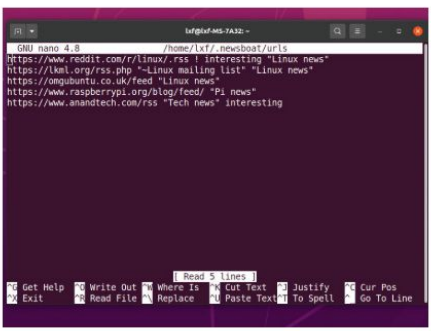
One of the standards that grew out of the early(ish) web was RSS (Really Simple Syndication) that was originally built into *Nescape Navigator* in 1999. It gave websites a way to announce updated content in a minimal way. Its popularity has come and gone, but RSS has stuck around even after *Google Reader* was killed off and the competing ATOM standard was launched.

An RSS feed offers a distraction-free way to obtain updates from your favourite website (although not **www.linuxformat.com** because that's just too advanced for us). If you combine this with the terminal then it means the latest news can be delivered directly to your favourite place to work.

We're going to take a look at *NewsBoat*. It's certainly not the only option, but it's currently actively developed, it's in most repos, there's a Snap (*you lot stop moaning – Ed*), it offers built-in HTML rendering, advanced filters, basic podcast support and loads more.



Here's Newsboat in all its basic reading glory.



A basic feed configuration showing how the tag system works with names and the Hide feature.

To install *Newsboat* on Ubuntu and Debian, type

```
$ sudo apt install newsboat
```

It's in both distro's repos, if you don't mind Snaps although it would be overkill for a terminal tool, then:

```
$ sudo snap install newsboat
```

If you want the latest bleeding-edge version then you can clone the Git, ensure you have all the build tools and dependencies, and then compile it from the source. Full details on how to do all that are in the online manual at **https://bit.ly/lxf280newsboat**.

Don't try and run it yet. There are no default RSS feeds and it'll kick out an error saying as much. We'll need to create a config file using:

```
$ nano ~/.newsboat/urls
```

At this point you'll want a few RSS feeds to add in. Perhaps we can help out with a few suggestions if you can't think of any offhand.

```
https://www.reddit.com/r/linux/.rss
https://lkml.org/rss.php
https://omgubuntu.co.uk/feed
https://www.raspberrypi.org/blog/feed/
https://www.anandtech.com/rss
```

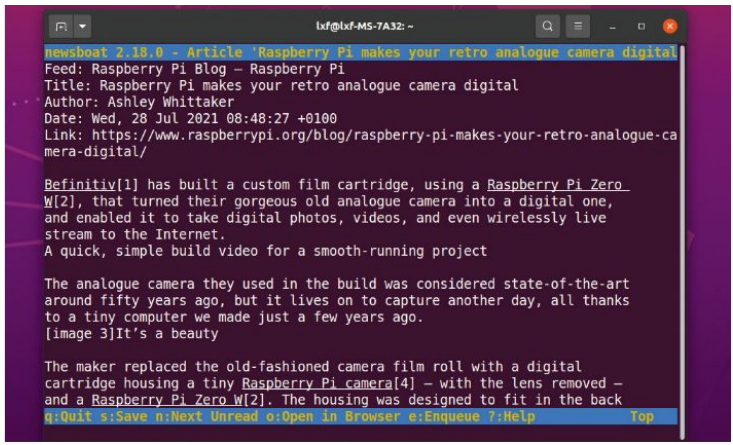
If you have a subscription RSS feed that requires a username and password to access the information, then the format looks like this:

```
https://username:password@hostname.domain.tld/feed.rss
```

If you're going to store passwords within the *Newsboat* configuration file it's best to ensure it can only be read by yourself with:

```
$ chmod u=rw,g=r,o= ~/.newsboat/urls
```

With a few feeds now configured start it with `newsboat -r`. Adding the `-r` forces *Newsboat* to refresh all the feeds. Why wouldn't you want this all the time? If you have a lot of feeds it's just wasting time and if you start *Newsboat* without the `-r` then once it's running you can individually refresh a feed by pressing r or refresh all feeds with R.



Basic navigation

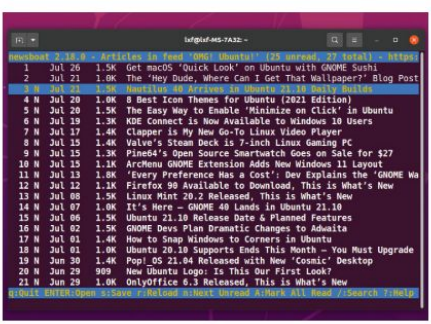
Once you're in *Newsboat* and have actual articles to read, navigation is pretty obvious and quick:

- **up/down arrow keys** – move the selection up and down by a single line
- **Enter** – read the currently selected article
- **q** – exit the current article/feed and then *Newsboat*
- **n** – jumps you to the next unread article when pressed while inside an article
- **Home/End** – jumps you to the start/end of either a list or an article itself
- **Pg up/Pg down** – jumps you a single page up/down
- **?** – lists all keyboard shortcuts
- **/** – opens the keyword search

Because the point of RSS feeds is simply to give you an outline of the article, if you want to read the full thing on the actual web site press o to open it in your default browser.

Adding tags

If you get really into RSS feeds then you can end up with a long list of them. To keep things organised it's recommended that you group feeds using one or more tags. So, for example, you could lump all the news sites into a "news" tag. It's no more complex than going back into the URL configuration file, leaving a space after the URL and adding the required tag name. If you want a longer tag that has spaces just enclose the tag name in "double quotes". You're able to use as many tags on a feed as you like – just separate them with a space.



The main article list is quick and easy to navigate.

When you now start *Newsboat* press the t key to view feeds organised by tags and use q to "quit" back to the normal feed view.

This simple tag system extends to renaming the feed name that's scrapped from the publisher. Again, after the URL give it any name that you'd like, starting with a ~ to show *Newsboat* that's the new name. Again, use double quotes if the name has spaces. If you want to hide a feed for some reason then use an ! after the URL – though it still appears in the tag view. You can also replace the ~ symbol with the ! as part of the name and it'll have the same effect.

RSS feeds are funny things. They certainly seemed to fall out of fashion at one point, but they're still around and can certainly help your scan and organise a heap of news and current events into a single, easy-to-access location. *Newsboat* is not the only option out there, but it's a quick and easy terminal option that's a worth a closer look. LXF

Most sites will only provide an overview of articles, but some will offer the entire text via RSS.

» RSS SERVICES

If you're already using RSS feeds via one of the many web services then good news – it's pretty easy to configure *Newsboat* to act as a newsreader for many of them. What's that, you ask? As you might have noticed, most RSS feeds are limited to a small number of current articles – 10 to 30 or so (the Linux kernel mailing list, or LKML, is unusual at 100+). So if you're not reading every day then it's easy to miss articles over time. Another option is that it means you can keep your read article list synchronised across multiple devices because they're all synchronising from the same online service.

Some online RSS services that are compatible are **theoldreader.com**, **newsblur.com**, **feedhq.org** and **bazqux.com**. There are additional self-hosted options supported, but this is something that's outside the scope of this article.

You should read the documentation for specific configuration of each service, but you'll need to add this to the configuration file:

```
urls-source "service name"
feedhq-url "https://serviceurl.com/"
feedhq-login "username"
feedhq-password "password"
```

VIRTUALBOX

Credit: www.virtualbox.org

Maximise the power of VirtualBox networking

Stuart Burns builds upon the powerful components of advanced VirtualBox networking with this second instalment of stress-saving tips.

OUR EXPERT

Stuart Burns is a Fortune 500 network administor specialising in virtualisation at scale. When not doing that he can be found experimenting with anything technical.

Last month we covered how to build an isolated but internet-capable test network for experimentation with no risk of breaking your network. Assuming that everything is working as expected, this month we'll build on what was created in the last issue (*see* **https://bit.ly/lxf280network**). We'll create a more complex network environment incorporating additional virtual networks and add additional physical hosts within the internal-only environment that comes with VirtualBox.

It can be useful to have more than one test network, given that Linux users like to experiment with multiple technologies or even mimic a larger, more complex network. The networks can either be isolated or joined to the existing infrastructure.

By default, *VirtualBox* has only one internal-only "intnet" network to select out of the box. For this tutorial's purposes we require an additional network. Fortunately, it's straightforward to add additional host-based virtual networks to the *VirtualBox* environment.

To add a second internal only network, create a new machine (or open an existing one) and navigate to the *VirtualBox* network configuration page for the VM (virtual machine). Select Internal from the Attached to part of the interface type. Where the name field is set to intnet, just use the cursor on the text field and type `intnet1` or whatever name is appropriate. This creates a second isolated internal network to use.

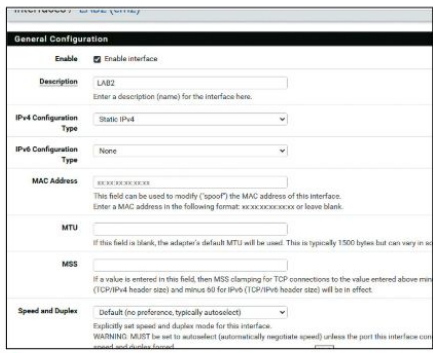
That network still has to be added to our new pfSense network and enabled in *VirtualBox* and the pfSense router. To do this, power down the pfSense router. Do it through the WEB UI, which ensures the shutdown is clean. Click the Diagnostics dropdown and select Halt System. Then edit the pfSense VM in *VirtualBox* and navigate to the VM Network tab.

Select the third network adapter (use the PDF above to follow from the previous instalment where we created an upstream network and an isolated network) and enable the network interface using the tick box. Be sure to select Internal network and the select the second intnet network that was created a few moments ago.

At this point the third network is attached and its IP configuration still needs to be enabled and configured.

Boot the pfSense VM up. Using the first VM we created, on the first lab network, open the web browser to log in to the pfSense GUI. Navigate to the Interfaces tab at the top of the GUI webpage. There'll be a new network listed, called (OPT1). Additional networks are not enabled by default. To enable the new NIC, just tick the Enable Interface box and save the configuration. If you were to save and reboot at this point, the pfSense console would show that third network.

OPT1 (Optional Interface 1) is a rather boring and not particularly useful name. It's easy enough to change, though. Go to Interfaces, Assignment and click the green add button for OPT1 NIC. This loads a new page with various details. To rename the interface, just click OPT1, then edit the Description file to suit.



If you don't use firewall rules you won't obtain the expected results. Remember to click Apply once the rule has been created.

A static-state network

There are additional options that need to be configured to make the second network useful. The IPv4 configuration section of the OPT1 network page is important. It's here that items such as IP address and DNS mask can be edited. It's possible to use either static or DHCP for IP configurations, but in this example we're going to have a pure static network. This means that every VM will need a static IPv4 address. From the options on the page, select the Static IP option. Be sure to make sure the interface is enabled too, as shown in the screenshot (*above*).

QUICK TIP

PfSense has many more goodies hidden away, ready to install. Useful examples include snort, syslog or NMAP. These tools aka "packages" can be found under x and y. Select them and they will install themselves. Exercise caution, as some may require significant resources.

Change the "Static IPv4 Configuration" portion. This is as simple as creating a new gateway address that will reside on the pfSense router (192.168.2.1 in our example.) and a subnet from the drop-down menu. (Usually a /24 is most useful). In addition, remember to check that Block private networks and block bogon network options are not ticked (or the traffic will never reach where you intended it to).

If DHCP is desired it can be enabled by going to "Services/DHCP server/Interface name" and select the tick box for the additional DHCP scope for the second network. If you wanted to get a bit fancy, there's nothing to stopping you from creating a new local-only DNS server to resolve the requests for that network and use another DNS for production networks.

Now you have the opportunity to decide how this second network is to be used. There's no connectivity out of the box for the new network, so it's time to correct that.

Set up a firewall

By default the LAN (our first network we used last time) can talk to any other network (as defined in the rules for that interface). Traffic can enter from the LAN. It's easy to modify the allowed traffic in and out by opening the pfSense GUI, navigate to Firewall/Rules/OPT1 and click Add from the bottom of the page. This opens the firewall rules addition dialog. For example, to allow HTTP traffic in, select Action and set it to Pass, set Interface to OPT1, set source to LAN Net and destination to OPT1 Net. Set the protocol to HTTP. This simple example enables the first network to talk to the second network on port 80, in other words a web server. Obviously, this can be configured as required.

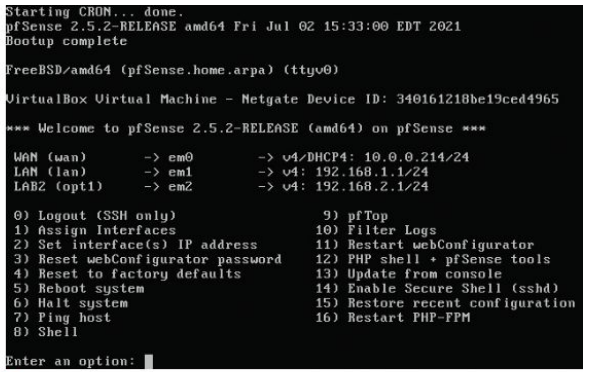
This is all very well, but the one snag is that this second network has no internet or DNS access. Fortunately, this is correctable (if so desired). Repeat the process of changing the network, but select the destination to be Any. You'll need to set up a route on all the machines that requires access to this second network, via the pfSense router. As an example, to enable a local network to access the new network, use the following command as root:

```
$ sudo ip route add 192.168.2.0/24 via 10.0.0.241
```

The first network part of the command is the network to reach and the second is the pfSense router on the "real" local network.

One topic that few people consider is using a second external computer and adding its resources into the *VirtualBox* infrastructure. Imagine your local desktop has a network with several VMs on it, but has run out of resources. You could create additional capacity by using a physical connection to the second box, but you don't want them sitting on the main network.

The solution is to add an additional physical host to the infrastructure, as long as you have a spare Ethernet port on the workstation or laptop. If you plug in the second host, the network will be shown in *VirtualBox*. This will allow the addition of extra capacity to be added to the virtual environment without it appearing on the main network (unless desired). The interesting thing is that it doesn't even have to be another *VirtualBox* host: it could also be ESXi, Hyper-V or whatever is desired.



The pfSense console, showing all the networks. This can be useful as the networks are listed on the console page, making troubleshooting easier.

To add in this capacity, add a fourth and final network interface on the pfSense network. Open the network on the pfSense box, choose networks and from the selection select Bridged adapter. Select the appropriate NIC from the NAME drop-down. This will essentially function as a third lab. To enable access, repeat the process of enabling the fourth network adapter in pfSense and creating appropriate firewall rules. Obviously the IP configuration for the physical host will need to be changed to sit on the correct network range for the new lab network.

At this point you should be able to create a fairly comprehensive test lab including physical and virtual hosts. Note that this internal setup isn't geared towards security and be cautious about what's exposed to the greater outside world. Keeping it internal shouldn't be an issue. Finally, use pfSense's backup export tool to save a copy of the current configuration. LXF

» INSTALLING VIRTUAL TOOLS

Out-of-the box virtual machines that are installed in a *VirtualBox* environment use some pretty non-performant but safe base configurations, because the virtual drivers aren't available in the OS base image. Installing the *VirtualBox* drivers is important. For Linux hosts, navigate to the VM console in question. From the *VirtualBox* console menu click Devices and select "Install Guest Additions CD image" to mount *VirtualBox's* tools. Mount the image (`sudo mount /dev/cd0 /media`) and run the **LinuxAdditions.run** file. *VirtualBox* should now run through the installation of the performant virtual drivers.

On Windows, repeating the same process of mounting the image will mean the CD will show up as a CD . Navigate to the CD and find the **VBoxWindowsAddtions-setup64.exe** file to install the tools.

Sometimes however, the Linux install will require additional items including header files, gcc and *make*. Install them using `sudo apt update && sudo apt install build-essential linux-headers-$(uname -r)` this installs all the dependencies required to make the Linux VM tools install work correctly. Usually installing gcc, *make* and the appropriate Linux headers and re-running the installation will work. If you run pfSense under VMware there's an **open-vm-tools** package that can be installed to boost performance.

XIBO

Credit: https://xibo.org.

Build a custom digital signage system

Discover how to create a digital signage display for an open source conference using Xibo digital signage and the help of **Matt Holder**.

OUR EXPERT

Matt Holder has worked in IT support for over a decade and has always tried to utilise Linux alongside the other installed systems.

Most *Linux Format* readers will have noticed an increase in the number of large-format screens in public places, displaying a range of information. This is called digital signage. In this tutorial we'll be looking at *Xibo*, a flexible system that can be used to show images, videos, RSS feeds, weather data, mapping data, tables of information, websites, embedded HTML and more. We'll show how to set up a server and client, before finally designing a layout and scheduling this content.

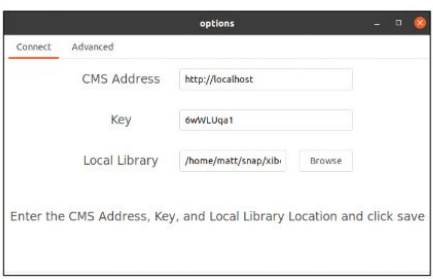
The system comprises a server component, which is open source, as well as a number of clients for different display types. The Windows and Linux clients are both free and open source, whereas the other platforms are paid-for, licenced clients. The server can be self-hosted, or hosting can be provided by the team themselves.

Before going any further, let's talk about some of the terminology utilised by the system. Displays is the name of the device which runs the player software and displays the content. This could be a Windows/Linux PC, Android device or large-screen TV running embedded software. Display groups can be used to group multiple displays and content can be scheduled to multiple devices at once. Schedules are then used to define when content should appear on Displays.

Layouts are designed by the end user to define what content should be displayed on specific areas of the display. Layouts are then split into Regions and each Region contains a playlist. See the diagram (top right), which explains the basic concepts of the *Xibo* digital signage system.

QUICK TIP

Using PHP functions, more advanced projects can be carried out. For example, if information is held in a dataset about meetings booked into a meeting room, then the current date and time can be used by the CMS to serve relevant content to the player. See https://community.xibo.org.uk/t/getting-started-guide-datasets/14149.



This information is required to configure the client.

Enter your username and password at the login screen.

Each Region can then display a number of pieces of content within its timeline. These concepts will become clearer as we work our way through this tutorial.

Installing the server

The *Xibo* server utilises a LAMP stack (which refers to the Linux operating system, Apache web server, MySQL database server and PHP programming language) as well as other tools to provide messaging between the clients and server. In recent years the entire setup has been wrapped inside *Docker* containers to make installation and updates as simple as possible. The server we'll use to install this on is Ubuntu 20.04, but container systems make the base OS fairly unimportant, so feel free to work with what you're most familiar with.

The software will work equally as well on bare metal or in a virtual machine (VM). When used in production, however, a VM wouldn't necessarily be the most suitable choice, especially due to possible issues with video playback using the VM's drivers.

Once the base machine has been installed, open a terminal, run updates and then install *Docker* and *docker-compose*. Commands should be run as root or prefixed with sudo:

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

```
apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
apt install docker-ce
curl -L https://github.com/docker/compose/releases/download/1.24.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

With the prerequisites installed there are a few steps required to configure the *docker-compose* files to enable the *Docker* containers to run and communicate with each other.

```
mkdir /opt/xibo
cd /opt/xibo
wget -O xibo-docker.tar.gz https://xibo.org.uk/api/downloads/cms
tar --strip-components=1 -zxvf xibo-docker.tar.gz
cp config.env.template config.env
nano config.env
```

On the relevant line, enter a strong password for the MySQL database and configure any other required options. Find out more about the *docker-compose* file here: **https://xibo.org.uk/docs/setup/xibo-cms-with-docker-on-ubuntu-18-04#create_config.env_file**.

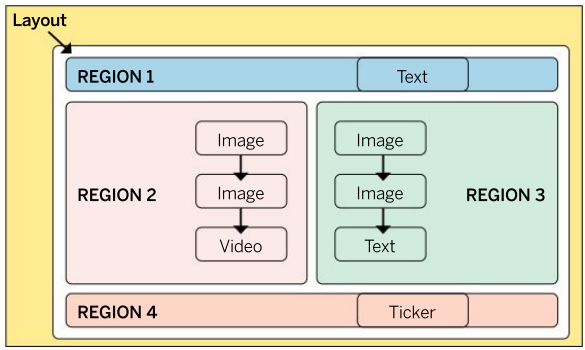
Next, bring up the containers using `docker-compose up -d` and allow *Docker* to complete its work. The server has now been installed and the server can be accessed from your favourite browser, by entering **http://IP_ADDRESS_OF_XIBO_SERVER** – enter your username and password at the login screen.

The player

There are open source players for Windows and Linux as well as paid-for options for other platforms as well, such as Android, Tizen and Web OS. In this article we'll install the player on Ubuntu 20.10. The *Xibo* team uses Canonical's Snap packaging system, which makes the installation of the client straightforward and updates are applied automatically. Again, back in the terminal the client can be installed by entering:

```
$ snap install xibo-player --channel=stable
```

To enable the client to communicate with the server, there are a couple of steps that need to be followed.



This diagram explains the concepts of the Xibo digital signage system.

First, in a web browser, login to the CMS using the default credentials of `xibo_admin` and `password`. Navigate to the Settings tab and copy the key from the CMS Secret Key field. The player can be opened from the Desktop or from a terminal, by entering `xibo-player`. The first time this is opened, the configuration screen will load. Enter the URL of the CMS (server) component, paste the value from the CMS for the Key and a location that the client can use to save files to (see screenshot, *left*). The information shown in the screenshot is required to configure the client.

When selecting the Save option, the client will contact the server, which will report back that the Display isn't currently registered. Now, back in the web browser navigate to the Displays tab and one entry will be shown. On the right-hand side of the screen select the Options menu and then click Authorise. Finally, back at the client click Save and the client will report that the Display is active and ready to start. The options screen can then be closed and when the client is opened again the default layout will be displayed (see screenshot, *previous page*). Once configured, the client will download the files required for the default layout. These will then be displayed.

Conference time

The example that we'll be walking through are displays that could be set up around a conference venue. The layout (see diagram above) will contain a title in Region 1, text in Region 2 which will relate to videos and images

QUICK TIP

The forthcoming version 3 of Xibo contains lots of exciting features that are being developed, including interactivity, folders to store media, reworking of permissions and support for sending commands via RS232.

» LINUX PLAYER OPTIONS

Once the initial configuration of the client has been carried out, further options are set from the CMS. To access these options, navigate to the Display Settings and either change the relevant profile or create a new one, which can be assigned to a particular Display. Within the Display Settings a range of features can be changed, such as the interval between the client collecting information from the server, the size and position of the player window (the default is full screen) and whether shell commands can be executed by the player. The latter option is useful, because a display can be set to shut down at a certain time of day.

When the client is loaded, press the I key and then the information screen will load. This provides access to diagnostic information such as media that's being downloaded, if there are any invalid files and gives the option to exit the player without it being reloaded automatically. Because clients are devices that need to run for large proportions of the day, should an error occur then the default behaviour is for the player to reload itself automatically.

Pressing I when in the client brings up a status screen.

in Region 3. Region 4 will contain a scrolling ticker of an open-source RSS feed.

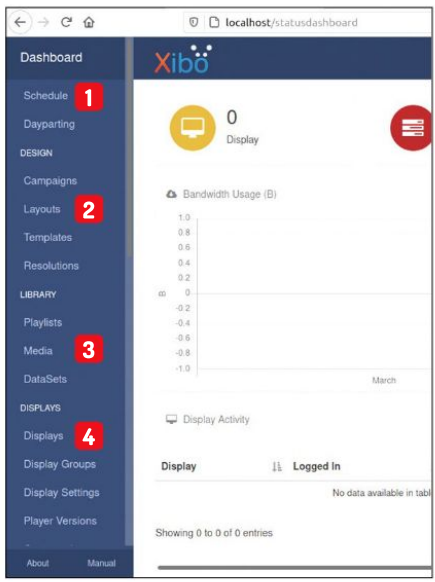
Before creating the example, we should investigate the GUI (*see below*). 1. Layouts are scheduled at a date and time. 2. Layouts are created here. 3. Media is managed here. 4. Displays are managed here.

Back in the web browser, login to the *Xibo* digital signage CMS, navigate to the Layouts section and create a new one. Once a name has been entered, the defaults should suffice. When created, the Layout editor will open. The first step is to resize the default Region, which is created by default. Select the option to edit Regions, which looks like an image of a pencil and notebook, and use the drag handles to resize the orange rectangle. Add further Regions to the layout, so that the image in the diagram has been replicated and four Regions exist within.

Regions can be set to loop, which is useful when they contain certain types of content. To do this, select the Region and then read the guidance before deciding whether to enable the loop option or not. Before saving the changes that have been made to the layout, set a light background colour or upload a background image by using the options in the right-hand panel.

Before carrying on with the setup of the digital signage, it's a good idea to take stock of what has been accomplished. So far, the server/CMS has been installed, the client has been installed and configured to communicate with the server, and the basic Layout has been set up with the correct number of Regions. In the next step, the Regions will be populated with information and finally the Layout can be scheduled and then viewed on the player.

There are four key areas in the CMS' interface (see below for details).



1 Layouts are scheduled at a date and time to show on a display.

2 Layouts are created here.

3 Media is managed here.

4 Displays are managed here.

Establish your Region

To start with, the top Region will be populated with a title. Add the title into Region 1 by selecting the icon that shows four squares, click the plus on the Text widget and then select Region 1 in the playlist editor. Select the Edit icon on the text item and enter a name for the conference as well as formatting the text. When completed, select Save (see screenshot, *below right*). When editing Regions, options appear on the right-hand side of the preview. This is an example of the options associated with the text widget.

The next step is to add an RSS feed of Linux news to the bottom Region. This is very similar to entering text, but requires the usage of the Ticker widget. This widget is formatted from the configuration panel in the top-right of the Layout editor. The defaults should suffice on the General tab. On the Configuration tab select the Feed URL field and add the link to a suitable RSS feed. On the Templates tab select Title Only and finally, on the appearance tab, select Marquee Left. When these changes have been made, save the changes.

One of the many powerful things about *Xibo* is the ability to schedule content in two different Regions that can change at the same time. This will be useful for the final part of this tutorial because we're going to populate the final two Regions with textual information on the left-hand side and a video and image on the right-hand side. The textual information will relate to the video being shown. When the video changes to a second piece of content, the text should also change at the same time.

The content that we'll be adding is a video of Big Buck Bunny as well as some information about it. The second piece of content will be a photograph of Linus Torvalds, as well as some information about him and the

» DO MORE WITH XIBO

As well as the media formats that have been discussed, *Xibo* can utilise a large range of resources to create effective digital signage displays. Audio files can be loaded as background music, calendar items can be read from, for example, a Google Calendar, countdown times can be displayed and clocks can be added to alert the user to the current time.

Looking at more data-oriented items now, currency trackers can be added as well as displaying data from the tabular dataset objects. These data can be displayed by means of a ticker, table view or chart view. Information from other websites can also be embedded using the Embedded HTML widget. When using this though, make sure you have permission to display information from another source.

While not strictly a media type, playlists are an incredibly powerful feature as these enable a previously configured timeline of items to be added to multiple layouts. When multiple layouts are scheduled at the same time they can either play sequentially or one can be configured to have priority. In this way an urgent layout can be displayed for a short period of time before normal scheduling is resumed.

Campaigns are used to group together a number of layouts and these can then be scheduled in one operation, rather than having to spend time scheduling multiple layouts separately.

Linux kernel. First of all, download a Creative Commons licenced photo of Linus Torvalds as well as the Big Buck Bunny film from **https://download.blender.org/demo/movies/BBB**.

To add the video, select the Region you wish to display it from and from the toolbox at the bottom hover over the video button and then use the Grab option to drag it to the Region you wish it to appear in. At this stage it's important to know the length of the video file, so that you can set the timing for the text in the adjacent Region to be the same.

Once configured, the client will download the files required for the default layout. These will then be displayed.

Add text to your display

To add some text, grab some from a suitable source and store it temporarily in a text file. To add this to *Xibo*, select the Region from the Region editor and then navigate to the Text button. Again, use the Grab button to drag this to the relevant Region. When the Text widget has been added, set the duration to be the same as that of the video file and click the pencil icon on the left of the Options panel. Paste the content into this Region and then format appropriately, before saving the Region settings on the right-hand side. Add a marquee effect to the Region if you want the text to scroll.

These previous steps can now essentially be repeated. With the Layout design tool, select the same Region that contains the video of Big Buck Bunny. From the Widget selection tool at the bottom of the screen, select Image and use the grab handle to drag this to the selected Region. When asked, select Add Files to select the photograph to be uploaded. When a thumbnail appears in the Upload media window click Start Upload. Now click Done and you'll see that in the Region in question there are now two items: a video and an image (you may need to scroll to the right). Select the image, click 'Set a duration' and set this to 60 before saving the Region.

Now select the adjacent Region and use the grab handle on the Text widget to drag it to the Region. This Region will then contain two text widgets. Make sure you check that the text widgets appear in the correct order and therefore match the order of the content in the other Region. If necessary this can be changed by dragging the order. With the final text widget selected, set a duration of 60 again and then use the pencil icon on the Region preview to enable Edit mode, where you can paste in your content and format. As before, set a marquee effect if the text should move in your Region.

Now that the content has all been added to the layout, this can be previewed within the web browser. To do so, click the back arrow button, which is underneath the Region editing preview. This will then display an overview of the entire layout. Simply selecting the Play icon, which will show a preview in the browser. Please note that not all content can be previewed in this way.

Once the layout has been completed, the final stage is to use the Actions option at the top of the screen to publish the Layout. This means that the changes will be saved and ready to be displayed. When making further changes to the layout, these can be saved part-way through and then published at a later time. Also, publishing changes can be made on a schedule, which can come in very handy.

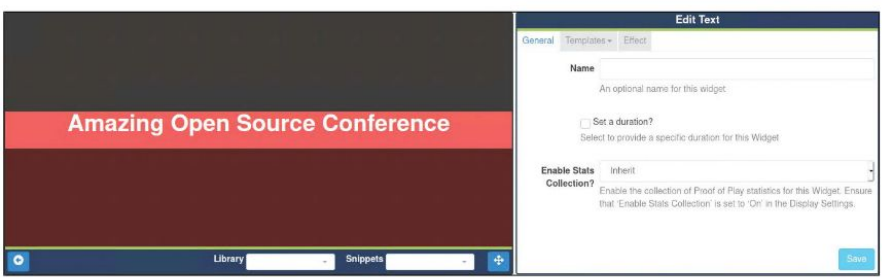
The final stage in the process is to schedule the Layout to the display. To complete this, on the left-hand side select Schedule and add a new entry. When the pop-up loads add the Layout, select the Display, Layout and select the date and time options for the Display. Use the option to save the schedule addition and this can be seen on the calendar (make sure to select the Display in the option at the top of the screen if the calendar appears empty).

Now, moving back to the Desktop, open the *Xibo* player and within a couple of minutes the content will be downloaded and displayed on the player.

So far we've merely scratched the surface of what this digital signage system is capable of. For some more advanced things to try, look at using datasets to display table data (datasets can also be used in a ticker) and the use of PHP functions to display certain parts of the dataset data, based on the date and time. For further information, the *Xibo* team has spent a considerable amount of time crafting documentation to support all users with the software. This can be found at **https://xibo.org.uk/manual/en/index.html**. LXF

QUICK TIP

The default credentials for the server are xibo_admin and password. The password should be changed after first login and can be done so by selecting the avatar in the top right-hand corner and selecting Edit Profile.



When editing Regions, options appear on the right side of the preview. This is an example of the options associated with the text widget.

30 YEARS OF LINUX

HotPicks

VLC » Ksnip » Waterfox » Far2l » G'MIC » OpenSnitch » WinXP » OpenTTD » 0 A.D. » Lutris » KDE Connect

Alexander Tolstoy comes up with a special series of *HotPicks* to celebrate 30 years of Linux in this issue

Welcome

The original issue 1 HotPicks, still going strong 21 years later!

Welcome to this special edition of *HotPicks* dedicated to 30 years of Linux! As usual we're covering the best open source software on the planet, but for this issue Alexander Tolstoy travelled back in time to revisit past issues of *Linux Format*, searching for the very best applications that proved to be the most useful and well-loved by the people.

Revisiting early *Linux Format* issues is fun, educational and sometimes nostalgic. Many well-regarded software titles from the mid-2000s are no longer maintained and appear to have passed away. Many personal open source projects, which used to be hosted on self-made websites or public code hosting servers, no longer exist (who remembers BerliOS?). Even GitHub, which is the main place to find a hot cool tool these days, wasn't around during the early *LXF HotPicks* issues: it emerged in 2008 and it took several years for it to become mainstream.

Of course, there's another camp of venerable open source picks that have long records of development, but these are mostly large projects with full-time staff and generous funding. We've always tried to avoid promoting high-profile Linux software such as *Firefox* or *LibreOffice*, since they already enjoy worldwide popularity. The ideal *HotPick* would always be from the underrepresented group of small projects that are looking for a wider audience. Such a *HotPick* pick also had to be of high quality and proven practical use, which further narrows our search.

As such, a nonpareil always implies a solid balance of several factors. An exclusion of this rule is a small group of entertainment apps. These are good-humoured programs, ideal for sharing with friends and colleagues. From the hilarious *Hollywood* hacker screen (**LXF195**) to the prank-like *CommandLineHeroes* fun testing tool (**LXF276**), they all help to raise the profile of open source software. We wish there were a bit more of such projects available for the Linux audience. For instance, there's the irresistibly charming *Desktop Pet* (*eSheep*) (see **http://bit.ly/2UuDd5k**), but unfortunately it's a Windows-only thing.

We hope you will enjoy this special edition of *HotPicks* and stay with us for more great FOSS finds!

» TOO HOT TO HANDLE

All of our picks this month are actively maintained open source projects with large audiences and constant user-provided feedback. The question is: what's the best way of trying a *HotPick*? Not that it was unanswered before, since we've published a comprehensive tutorial on building various software out of its source code (see **LXF256**), but this is a good opportunity to provide a few more tips on the topic.

Modern Linux distributions normally have huge software repositories made up of thousands of packages. There's a good chance that the program you're looking for has already been packaged and is provided through the official channel of your distro. If not, check if the developer provides a ready-to-run download option under the Releases section on the GitHub page. Sometimes there's just an archive that you're supposed to unpack and run the program from, which is the case for *Waterfox*, in one of our greatest *HotPicks* this month.

Still no luck so far? Try to find a Snap, a Flatpak or an Appimage of the software. These formats represent three main ways of delivering a sandboxed version of a Linux application. As for Snap and Flatpak, you'll need to have appropriate back-ends installed on your system.

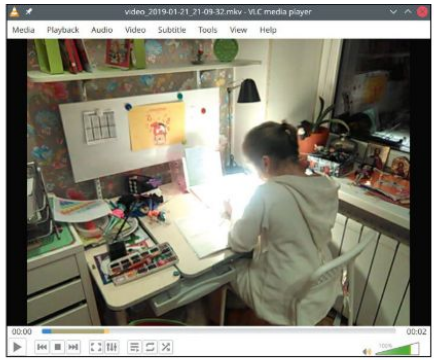
If you're running a Linux distro that doesn't have a store for such sandboxed programs, then don't miss the outstanding Bauh software store (see **LXF266**), which enables you to explore, install and manage applications in any of the aforementioned software formats. And remember that building something from source isn't as scary as it sounds – it's usually nothing more taxing than typing a couple of commands.

Ensure you have build-essentials. Compiling from source should work... sometimes, probably!

BEST MEDIA PLAYER

VLC

Version: 3.0.16
Web: www.videolan.org/vlc



VLC will be familiar to the majority of computer users, not just Linux fans.

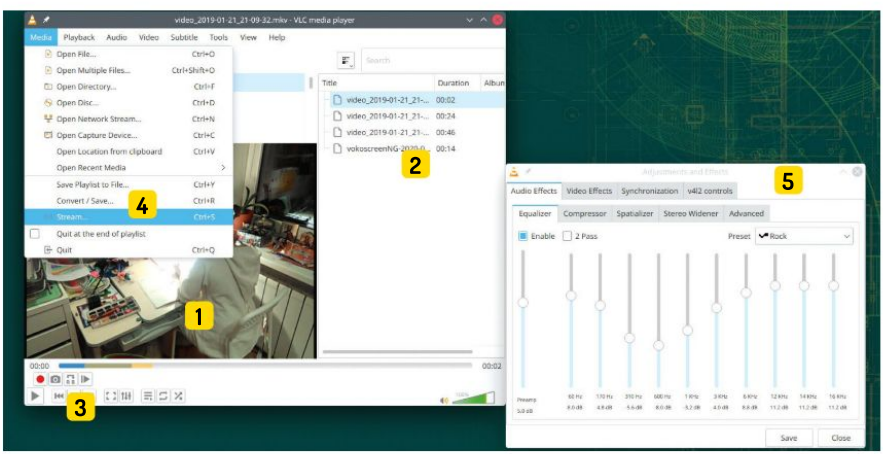
We think there's no real need to have different categories for music and video players. Most video players are more than capable at playing music as well. At the very least you can add a selection of local music tracks and manage your playlist. It's hard to please everyone, but we've discovered that it's the *VLC* player (see **LXF236**) which has the most number of devoted Linux users.

This is a hugely popular video player that combines several attractive features. First, it sports a clean, intuitive interface, despite the fact that there are a wealth of settings, options and configuration menus for power users. For mere mortals it's easy to use *VLC* for basic playback with minor adjustments where applicable: for example, changing the audio track, choosing a custom equaliser preset or rearranging items in the playlist. Second, *VLC* already comes with full multimedia format support. You don't need to check if your system has a specific codec because it's already bundled with *VLC*. Furthermore, the player automatically uses GPU acceleration where possible. Some time ago we reviewed *Qmplay2* (**LXF191**), another video player that takes a different approach: you had to manually select the VDPAU or VAAPI video back-end in its settings. It's still not hard to do these days, but *VLC* strives to offer the best experience right out of the box.

Another area where *VLC* excels is streaming. First of all, it can easily handle online streaming sources, such as radio stations. Next, it's straightforward to stream your own content to LAN or to remote hosts on the web. Literally everything that can be played back – files, discs, webcams, network streams – can be streamed or rerouted to another location. After selecting Media> Stream you'll be guided by the streaming wizard that explains everything perfectly.

VLC serves both the pro audience and newcomers. The player is included with most Linux distributions, so you can have the program up and running in no time.

EXPLORING THE VLC INTERFACE...



1 Wall-to-wall codecs
VLC supports all the popular media formats right out of the box – no extra codec installation required.

2 Handy playlist area
It's hidden by default, but you can invoke the playlist at any time. Sort video and audio files any way you like.

3 Playback controls within easy reach
Here we've enabled some additional control buttons for starting recording and taking screengrabs.

4 Stream content with ease
Everything that VLC can play back can be utilised for streaming – even other streams. Use the Media menu to re-encode your files.

5 Advanced audio and video adjustments
VLC includes an equaliser together with a wide variety of audio and video filters and effects for any scenario you can think of.

BEST DESKTOP ACCESSORY

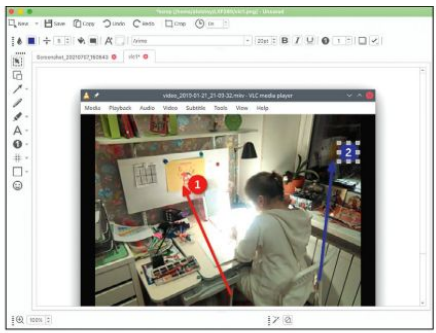
Ksnip

Version: 1.9.0
Web: https://github.com/ksnip/ksnip

A desktop accessory is usually a small application with a graphical UI that makes our daily Linux computing life easier. That's perhaps the most popular category in the history of *HotPicks*. These tools are usually one-person projects and therefore not all of them survive in the long run.

We've selected one application that's been actively maintained for several years and has grown from a tiny hobby project into a widely used Linux program. This is *Ksnip* (see **LXF266**), the screenshot taker for Linux. While pressing PrtScrn enables you to capture your screen, there's only basic software behind this and you'll be missing out on most of *Ksnip's* extra features. Indeed, the program is much more than a tool for taking pictures of your screen. A better way to describe *Ksnip* is that it's an advanced clone of the *Snipping Tool* from Windows. As such, it can capture specific parts of the screen while assisting you with a magnifying glass and a snipping cursor for pixel-perfect precision.

The main benefits of the program become apparent after the screen is taken and your picture is loaded into



No other software is as good at annotating images as Ksnip.

the *Ksnip* image editor. This contains an abundance of annotating tools that can achieve much more than a *Paint*-like image editor. For example, *Ksnip* enables you to paint auto-numbered circular tags that save a lot of time. The only other software for Linux that can do this is *Shutter*, which has long been unmaintained and has only started to recover recently. The *Ksnip* image editor can also blur selected regions to help you protect personal or sensitive data, and also makes it possible to draw arrows, rectangles, add text captions and more. Most of the tools enable you to change various options, such as colour and size.

Ksnip has overtaken Microsoft's original *Snipping Tool* and is now a cross-platform program available for Windows as well. In the Linux world it remains its most advanced and feature-packed screen grabbing tool.

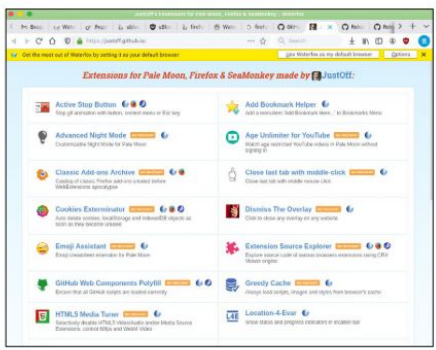
BEST WEB BROWSER

Waterfox

Version: G3.2.3 **Web:** www.waterfox.net

A quick guess tells that the winner of best web browser should be either *Chromium* or *Firefox*, but after sorting stacks of open source web browsers we excluded underdogs like *Falkon* or *Otter*, and were left with a fabulous *Firefox* spin-off called *Waterfox* (**LXF272**). This browser retained everything we loved in the stock *Firefox* offering, but added better privacy and some options that were removed from *Firefox*, such as NPAPI plug-ins and legacy extensions. Although some old plug-ins might have security flaws, their removal from *Firefox* broke a lot of functionality of the existing web applications.

Catch up with the future, or find yourself left behind and unsupported? With *Waterfox* there's no such issue. It's the end user who decides if they want to use a legacy plugin or disable it. *Waterfox* isn't just a tweaked version of an older version of *Firefox*, and neither it is a *Firefox* ESR clone. The program receives security and feature updates and delivers a modern web experience powered by the proven Gecko 78 engine under the hood. In our tests *Waterfox* also performed much faster than *Firefox*. The former is less bloated, doesn't collect



Imagine Firefox with a more conservative approach and a range of legacy features – that's Waterfox!

telemetry and doesn't track users (although limited data collecting may still occur on certain sites).

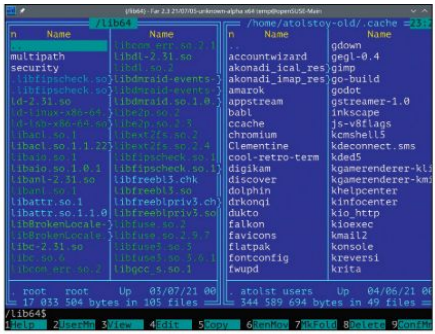
Another *Waterfox* performance optimisation is possible because it uses the Intel C++ Compiler (ICC) instead of GCC – at least, that's what you get when downloading the Linux x64 pre-compiled binary build. Even though the significant share of the browser's code is Javascript, it still heavily depends on the performance of C++ code, and it really shows when loading resource-intensive websites.

Furthermore, almost all *Firefox* features and add-ons are compatible with *Waterfox*. You can get any *Firefox* extension to work, be it a modern web extension or a legacy XPI file, and you can also use Sync, Monitor and Send services by Mozilla in *Waterfox*, too.

BEST FILE MANAGEMENT

Far2l

Version: v2021-06-30_alpha
Web: https://github.com/elfmz/far2l



The quest for the best clone of Far Manager for Linux is over – here's the port of the original software.

Alongside the command line shell, file management is a key computing topic. Over the years of reviewing file managers for *HotPicks* we've identified two major groups of these programs. One replicates the original desktop metaphor introduced in early Apple Macs and made popular by Windows. That's generally what *Dolphin* in KDE, *Nautilus* in Gnome and *Nemo* in Cinnamon achieve. We've also reviewed some worthy alternatives, such as *KDFM* (**LXF278**), which is the only file manager that has the Cover Flow carousel view.

The other group consolidates various twin-panel file managers that inherit the legacy of *Norton Commander* and *Far Manager*. We've featured various clones of them here, including the ever-popular *Midnight Commander* and *Wal Commander*. In fact, the contest for the best clone of *Far Manager* could never end if only there were no official *Far Manager* version for Linux. And we have it! The project has existed since 2016 and still receives many frequent commits. *Far2l* (its shortened name) isn't just another deep-blue, old-school program for grey-bearded users. We explored many of *Far2l's* features in **LXF271**. In brief, it simply works better and more predictably in various non-standard cases. To name a few, you can expect *Far2l* to display correct filenames inside ZIP archives even if a different encoding has been used (such as OEM/DOS). *Far2l* knows how to handle password-protected archives and also how to open tar.gz files.

Yet perhaps its coolest feature is the *Netrocks* plugin that enables *Far2l* to access remote locations via ftp[s], scp, sftp, nfs, webdav and even Samba! Some basic multi-tasking is also supported: you can continue using *Far2l* while a background network operation is running. All this works in a pure console mode – no graphical environment required.

BEST IMAGING APPLICATION

G'MIC

Version: 2.9.8 **Web:** http://gmic.eu

Get creative in image processing with G'MIC and its mind-blowing set of over 400 effects and editing tools.

Debates over the best image editor for Linux show no sign of disappearing. In many, if not all software group tests that include *GIMP*, it's this venerable program that takes the top spot. However, it depends on what exactly we mean by 'editing'. It can be photographic retouching using the Stamp and Healing brush, or it can be drawing, selecting and working with layers. People often edit photos by manipulating their colours, fixing gamma, brightness, saturation, restoring over-saturated areas and so on. Looking back at such imaging tools that proved to be of the most use in real-world tasks, applications such as *Rawtherapee* (**LXF247** and **237**) or *Darktable* (**LXF273**) spring to mind.

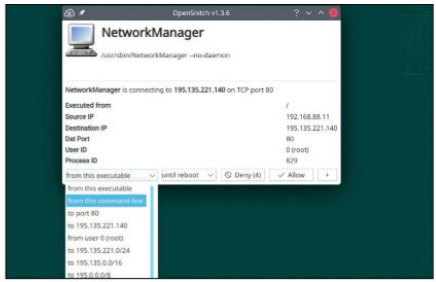
However, not everyone wants to manipulate RAW files. Luckily, there's an alternative called *G'MIC – GREYC's Magic for Image Computing*, which we looked at in **LXF263**. This is a collection of filters and effects that significantly lowers the entry barrier for newcomers. *G'MIC* is an easy-to-use graphical browser of dozens of image effects that are arranged as a tree. The tool is a plugin that can be attached to *GIMP*, *Krita* (**LXF237**), or used as a standalone editor.

G'MIC has a solid set of professional colour mixing, blending and editing tools that can be overlooked if you have a specialised RAW processor. If you're looking for deformation or abstract effects, contours, tiles, patterns, or maybe imitating oil paintings or sketches, you'll find them among *G'MIC* 400-plus filters and effects. It's the largest open source collection of image-editing tools that's available via a clean interface (CLI mode is also available). Compared to *Krita*, which ships with only a limited set of basic filters, *G'MIC* comes out on top. Each *G'MIC* filter usually has several adjusting sliders and there's a live preview of changes before you apply them.

BEST SYSTEM TOOL

OpenSnitch

Version: 1.3.6 **Web:** https://github.com/evilsocket/opensnitch



No outgoing network activity will be overlooked. OpenSnitch captures everything and lets you decide what to with it.

A system tool is an application that helps with system administering or managing resources. We've had plenty of such applications in *HotPicks*, and in our view, the most useful system tool is *OpenSnitch* (see **LXF250**). This is an application firewall for Linux that's designed to change people's attitude towards security, who don't want to get too involved. The reason is that network security at home and in the office is often as different as chalk and cheese. While system administrators take it very seriously in enterprise environments, desktop folks don't want to bother with extra setup on their home Linux systems (yet still they want to remain protected).

Tools like *OpenSnitch* are designed to make home users better realise the current network activity and remain in control over it. The application was clearly inspired by *LittleSnitch* for macOS and offers unprecedented control over your network traffic.

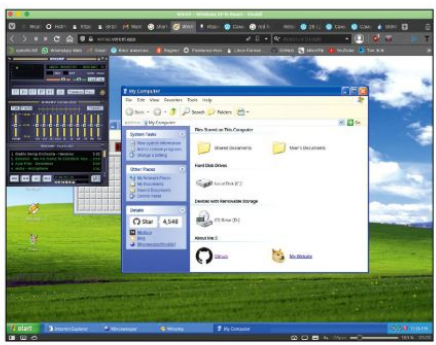
Here's how it works. First you need to get *OpenSnitch* running by installing and enabling its back-end (a Systemd service), and setting up the graphical front-end (a Python3 package). Once that's done and the program is running, you'll receive constant warnings from the *OpenSnitch* notifier. Whatever process in your Linux system wants to go online, it first gets caught and suspended by *OpenSnitch*. It may get annoying pretty quickly, but remember that you only need to set up the rules once, and after that *OpenSnitch's* interruptions will become less frequent.

However, the initial rules setup gives a unique insight into the hidden network activity in your system. You can see how the DNS resolver works, how the web pages you visit collect statistics, how the weather widgets try to fetch data, or even how a proprietary software tries to phone home (for example, *WPS Office*). For each case you can approve or decline a connection, and also make your decision temporary or persistent. By default, *OpenSnitch* is permissive, which means that it waits for a while to let you take action, then lets the traffic go.

BEST USELESS APP

WinXP

Version: GIT
Web: https://winxp.vercel.app



Your computer may be at risk! Oh, really? Windows XP is finally a web application, so get ready to play with it!

Some may wonder how a useless program can appear in *HotPicks*. In fact there have been several occasions when we discovered an application that was fun but had little or nothing to do with productivity (and wasn't a game, either).

It all started with Dustin Kirkland from Canonical, who wrote *Hollywood*, which imitated hacker screens (**LXF195**). A few follow-ups emerged, with the best one being perhaps the gorgeous *Edex-UI* (**LXF258**). Sooner or later any good activity simulator will mature, become populated with sensors, widgets, graphs, dashboards, and will inevitably be of some use. We definitely needed something different for this 'useless' section of *HotPicks*.

How about part of an operating system recreated as a web app? That's exactly what *WinXP* is, which recreates the Luna theme used in Windows XP from 2001. Keep in mind that *WinXP* is very limited: few desktop features are implemented, although it plays the Windows 'error' sound often enough. That said, when working on **LXF261** we didn't dare to review the whole *WinXP* web project and only went with *JSPaint* that time.

The only visible purpose of *WinXP* is to bring back feelings of nostalgia, with Microsoft Windows XP contained within a browser tab. You have the wallpaper, the bottom panel, the Start menu, the desktop icons and some links. You can drag and resize windows, browse the web in *Internet Explorer*, play *Minesweeper*, use *Notepad*, listen to the pre-defined tracks in *Winamp*, and draw in *Paint*. That's pretty much the extent of the program, since all other links say "Application not found'.

A closer look reveals that *WinXP* is just a setting for *JSPaint*, which is actually the only real application here. *JSPaint* looks and works exactly like *Microsoft Paint*, and even copies some bugs from the latter. *WinXP* will make a lot of people smile, or even wonder where they are if you press F11 in the web browser without them noticing!

BEST SIMULATION GAME

OpenTTD

Version: 1.11.2 **Web:** www.openttd.org

We realise that everyone has their own taste in games and therefore it's not possible to be fully neutral here. Still, we made some observations, tracked down each game's history, examined feedback, user replies and even closed issue on GitHub. Naturally, most small projects that just showcased their authors' coding skills went off the candidates list. We ended with two large, high-quality projects: *Julius* (**LXF233**) and *OpenTTD* (**LXF264**). Both will eat into your free time as you build, maintain and enhance cities. Both have huge armies of fans and plenty of user-created content to play and enjoy.

Another common feature between these two games is that they both recreate the gaming experience of two commercial games: *Caesar III* and *Transport Tycoon Deluxe*, respectively. However, *Julius* still requires you to have the original *Caesar III* game copy (it's abandonware, but still...), while *OpenTTD* has no such prerequisites – it's a purely open source project. Furthermore, *OpenTTD* has more elements and much more complex game mechanics than *Julius*, so we have to admit that *OpenTTD* has simply outclassed *Julius*.

The ideal sea port infrastructure is the result of many hours of perfecting each tile in OpenTTD.

So, what's so captivating about the transport business? Well, there's definitely something special about setting up routes and getting vehicles to run between stops. Your business grows and so does the revenue. Moreover, public transport boosts the growth of cities, which leads to more passengers, mail and goods. You can compete against other players that run their own companies, buy them or get sold, or try to beat the AI players.

Another area where *OpenTTD* shines is railroad transport. The game features the real-world (and therefore pretty complicated) system of semaphore signals and lets everyone take on the roles of railroad managers and dispatchers. Organising routine train journeys, avoiding collisions, enhancing the network throughput – these tasks remain entertaining for a very long time. And there are also ships, plains, trams, beautification add-ons and more!

BEST STRATEGY GAME

0 A.D.

Version: 0.24 **Web:** https://play0ad.com

Although *0 A.D.* is a tremendous time-killing game, it needs a different section to better fit the criteria of modern-class strategy games, with its high-resolution graphics and full 3D experience. *0 A.D.* (see **LXF275**) is the real gem in the world of open source games thanks to high standards of artwork, details and historical accuracy. The game originates from the full conversion mod for *Age of Empires II: Age of Kings* that dates back to 2001. Twenty years after that, *0 A.D.* is a completely independent and entirely open source game about economics and warfare.

The game invites you to govern one of the ancient civilisations, starting with the very basics. You have a camp and some workers that can build auxiliary buildings and collect resources. The goal is to develop a sustainable economy with sufficient resources (stone, iron, wood and meals) to produce military units and resist enemy attacks.

0 A.D. also has a multiplayer mode to let you compete with friends over LAN or online, but for newcomers it would be best to start learning the game in single-player mode. Play against the game's AI by building your own empire based on a chosen civilisation. In total there are 13 historic nations with their unique architecture, skills, names and other attributes carefully implemented in the game. The nations vary in their abilities: some may be better at defence or attack, or have unique large units such as elephants. The camp can grow into a town and then a large town giving you more abilities and the opportunity to build a bigger army.

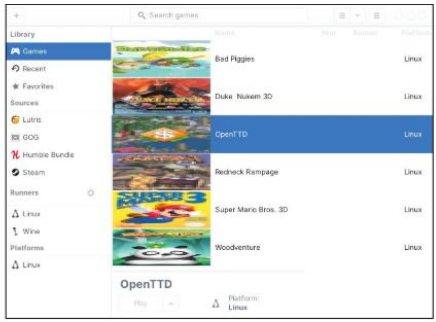
Our Romans are peacefully farming and managing cattle. But a battle with Egyptian armies is only minutes away from this rural bliss!

The game's mechanics are complex, but in the end it's a classic balance between the economy and war. One can't grow without another: you need to invest in technological advancements and care about building materials and population fertility to create good conditions for building defensive units. You'll need them – a skirmish is always just around the corner...

BEST GNOME APP

Lutris

Version: 0.5.8 **Web:** https://lutris.net



Collect all your games from various sources within this well-made organiser and software store.

There's no sense in dividing the world of open source programs into isolated groups, but Linux users tend to gather into several camps with their own preferences. One such division line concerns graphical toolkits. Gnome users would like every program to be consistent with the rest of the desktop: use GTK header bars and feel 'native' and 'integrated'. KDE Plasma users want the same kind of consistency for their desktop as well. There's absolutely nothing wrong in running a GTK-based app in a Qt-based desktop environment (*burn the heretic! – Ed*), but in this category we're looking for the best third-party application that would fit your preferred desktop environment, and we'll start with Gnome.

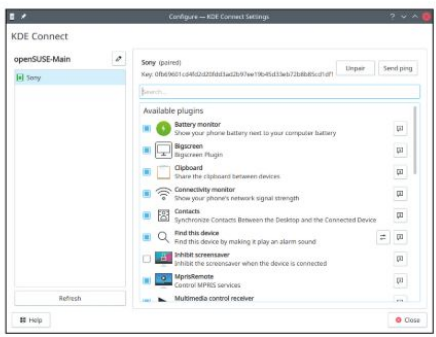
While we've looked at many great GTK3-based applications in *HotPicks* over the years, we needed to choose the very best one. It should be feature-rich and preferably the only one of its kind. That's exactly what *Lutris* is (see **LXF255**). Originally, we treated it as an organiser for games, but these days *Lutris* has grown into an open source software store. This is a neat GTK3 application that unites all your games under one roof. In particular, *Lutris* enables you install and set up various types of games once and then track your playing time, create game groups, manage favourites and more.

The best thing about *Lutris* is that it supports many different game runners. For instance, you may have various types of retro games that run natively on Linux, or use *Wine*, or *Dosbox*, or a Nintendo/PlayStation simulator, plus you may have a collection of more up-to-date games from GOG, Steam, Battle.net, Origin, Uplay, plus some bookmarked web games. You don't need to spend a lot of time trying to manually create a launcher for each game – let *Lutris* do this job. There are plenty of game runners for virtually everything that can run in Linux. Whatever your gaming tastes and preferences, *Lutris* can keep everything organised.

BEST PLASMA APP

KDE Connect

Version: 21.04 **Web:** https://kdeconnect.kde.org



Control your Android phone from Linux, or vice-versa with KDE Connect.

For a long time we were looking for an outstanding application that would have strong ties with the KDE Plasma desktop. Not just using the same UI toolkit or respecting KDE-specific environment variables, but something truly special.

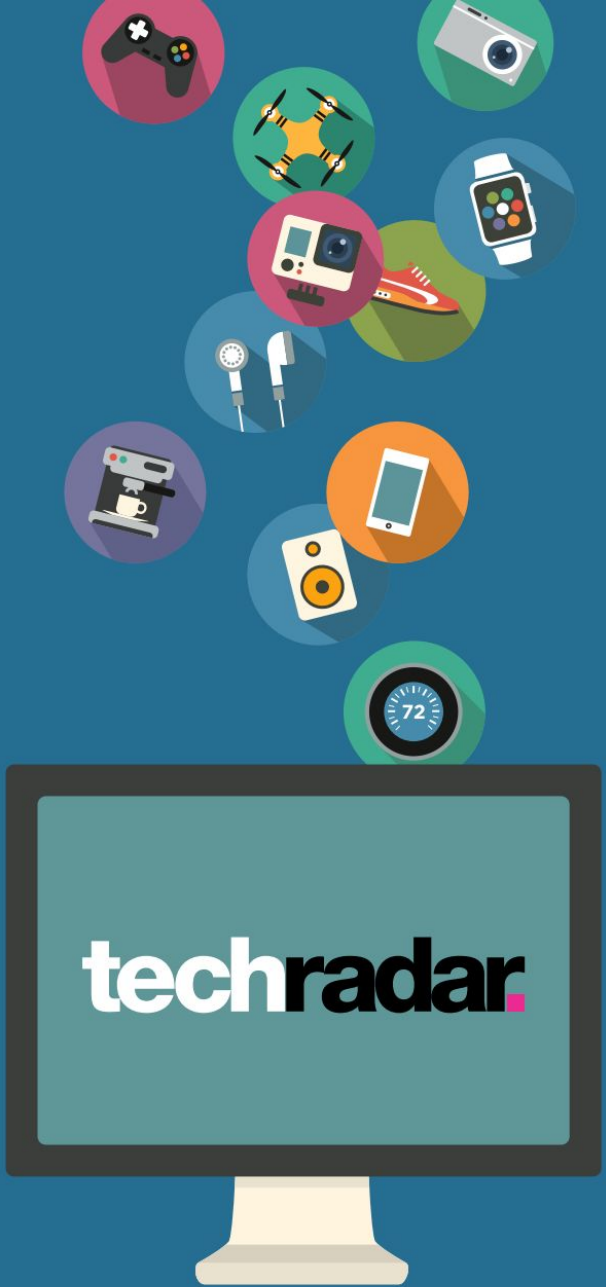
We've seen dozens of high-quality and promising applications using the Qt5 toolkit or even sporting a premier integration with KDE Plasma, but none was as grand as *KDE Connect* (see **LXF222**). Yes, we also have *GSConnect* (see **LXF268**) for Gnome, but the original software was created and tailored specifically for the KDE desktop.

KDE Connect is an undisputed winner this time because it adds a killer feature that wasn't available before. In particular, it links your Android phone with your Linux computer and enables sharing and remote control. Basically, it's how Apple had once integrated iPhone with its desktop OS.

There are so many things *KDE Connect* can do beyond simple notifications sharing that it'll take your breath away! Use your handheld screen for remote mouse control, access the phone's file system from Linux, control your desktop media player from the phone, read and write SMS, explore your address book, find your lost phone... the list goes on!

Install the *KDE Connect* application on your phone from the Play Market, connect both the phone and the Linux system to the same network, and finally make sure no firewall is blocking the connection. No extra steps are required on the KDE side since *KDE Connect* is normally bundled with it and available right out of the box (if not, check the package manager).

KDE Connect has always been the best in its class, and at some point it became cross-platform too. Users of macOS and Windows can enjoy the official *KDE Connect* versions – share the FOSS love! LXF



The home of technology

techradar.com

SECURITY

Build a software analysis Gitlab pipeline

Part Two! Missed part one of this series? Then turn to page 66

In the second part of our web application security series, **Tim Armstrong** takes us through the essentials of software composition analysis.

OUR EXPERT

Tim Armstrong is a former lead engineer turned developer advocate specialising in networking, software development and security. Find him on Twitter as @omatachyru.

Software developers the world over have a hard-enough time maintaining and securing their own code, so it's fairly common for the libraries and docker containers used, especially in large projects, to be a few versions behind. When was the last time you actually audited 100 per cent of the code for all of the software used in any of your projects. Never, right? You don't have time, you're not an expert in every language, and by the time you were done you'd need to do it all again. Software composition analysis (SCA) solves this problem by effectively doing this for you.

In this tutorial, you'll learn how to use a number of SCA tools to protect your code by extending the CI/CD pipeline created in the first part of this series, where we learned about static analysis and setting up a pipeline in GitLab CI. You can get a copy of where we left off by forking the repository at **https://gitlab.com/plaintextnerds/web-app-security-tutorial1-lxf279**, but we highly recommend picking up a copy of the previous issue and following that first if you can.

SCA tools such as *Snyk*, *WhiteSource*, *Gemnasium* and *Dependabot* scan your dependencies and containers for vulnerable versions, with the goal of either updating it for you via a pull request (PR) or notifying you of the issue. Each of them works in slightly different ways, uses different databases, and presents the data in different ways, so finding the right one for you takes some exploration. To this end, this tutorial will be looking at *Dependabot* which is open source, and *Snyk* which is reasonably priced and offers a free option for individuals and open source projects.

Snyk it to them!

Snyk is a hosted solution, so to get started you're going to need to create an account. You can do this by going to **https://app.snyk.io/login** and selecting the identity provider of your choice. There isn't a direct registration option, which could be an issue for people who don't trust any of the providers listed, but the selection is pretty big so it shouldn't be a problem.

Next, you'll be presented with the option to select the location of your source code. On this landing page the choices are GitHub and Bitbucket, but because this tutorial is using GitLab you'll need to click the full "list of integrations" link. From here you can select GitLab.

Setting up Snyk is pretty much a point-and-click process. Just give it some credentials and it gets on with it.

Because of the nature of what *Snyk* is doing, in order to get it working you need to give it a personal access token with API Scope privileges. This is the highest level of privilege that you can grant a token in Gitlab. So if you're working with confidential code (in other words, it's not an open source project) it's best to set up a dedicated account for it so that *Snyk* is acting as its own user. This is best practice when dealing with any third-party integrations in case the API key is leaked somehow and you need to identify unauthorised modifications easily.

To create a personal access token in GitLab go to **https://gitlab.com/-/profile/personal_access_tokens** (GitLab account required) in a new tab. Give the token a name in the Token Name field, check the box to grant it API scope, and click 'Create personal access token'. Confusingly this inserts an element (that contains the token) into the page just above the section where you entered the details. This only shows up once, so if you refresh or leave the page before you've copied it you'll need to delete the token and recreate it.

Now that you have your token, head back over to the *Snyk* tab, paste it in the box and hit Save. Now you've got it linked you'll need to add the project to *Snyk*, so go ahead and hit the button, which will take you to a page where you can select any of the projects the GitLab user has access to. Select your fork of the Web App Security tutorial code and click the 'Add selected repositories' button. This should find the **requirements.txt** file and start scanning.

QUICK TIP

For a refresher on setting up CI/CD pipelines in GitLab, check out the previous tutorial in this series, or head on over to https://blog.plaintextnerds.com where there's a dedicated tutorial series on GitLab CI/CD.

Looking at the results from the scan you can see that the version of Django used has a known vulnerability – specifically a SQL injection pathway known as CVE-2021-35042. This vulnerability was found in the time between the time of writing the previous tutorial and this one, which exemplifies the importance of having good SCA tools in your pipeline!

Plug the vulnerability gap

Hitting the 'Fix this vulnerability' button finds the smallest upgrade that resolves the vulnerability. If it's a major version (assuming semantic versioning – <Major>.<Minor>.<Patch>) bump then *Snyk* lets you know that there's a potential breaking change, so you'd better check the release notes.

Alternatively, you could wait for *Snyk* to create the Merge Request for you when it next does its scheduled scan. Every time it does a scheduled scan *Snyk* automatically creates a Merge Request (if one doesn't exist already) for any problems found.

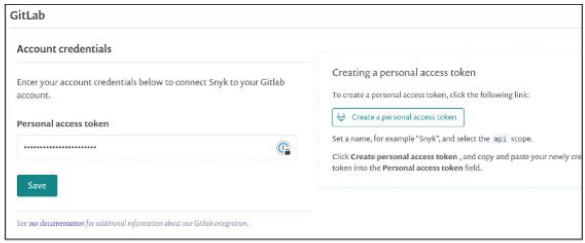
By default, *Snyk* scans your Repo once a day (and once per Merge Request) which is pretty helpful because it means that, unlike a pipeline workflow, the Merge Requests are still being created even if you haven't worked on a project in a while.

Hopping over to the Dependencies tab and hitting All Dependencies shows that not only is *Snyk* detecting the dependencies that you've defined, but also the dependencies of those dependencies (so-called transitive dependencies) by constructing a graph of each library's requirements. This ensures that you're also protected for issues that are deep in the tree. When using *pip* and **requirements.txt** this is more of a best guess solution though, so using a locking dependency manager like *Poetry* or *PIPEnv* can improve the reliability here by providing *Snyk* with all the information it needs to know exactly which versions you're using.

That old Dependabot

So *Snyk* is certainly useful, but perhaps you work at an organisation where giving a third party complete access to your source code isn't acceptable for whatever reason. This is where *Dependabot* comes in, because it can run as a standalone in your CI/CD pipeline where you can keep everything isolated.

A short while after the original *Dependabot* was bought out by Microsoft's GitHub, a GitLab-flavoured fork (**https://gitlab.com/dependabot-gitlab/dependabot**) was created by Andrejs Cunskis which has since been sponsored by JetBrains.



Creating a personal access token in GitLab is equal parts simple and scary. Most programs need API access, but with that comes risks.

There are a number of supported ways to get *Dependabot* up and running, but in this case you'll be needing the standalone mode because it has to be built into the CI/CD pipeline. To do this the first thing you'll need to do is create a folder called **.gitlab** in the project directory. In that folder create a file called **dependabot.yml**.

For this project, the minimum that you'll need to define in the **dependabot.yml** file is

```
version: 2
updates:
 - package-ecosystem: "pip"
   directory: "/src"
   schedule:
    interval: "daily"
```

While required by the file spec, the `schedule` directive isn't going to limit the run to once a day when using *Dependabot* in the Ci/CD pipeline (despite being set to `"daily"`).

Next, you'll need to provide *Dependabot* with a Personal Access Token with API scope credentials –it's exactly the same process as with *Snyk*. You'll need to go to **https://gitlab.com/-/profile/personal_access_tokens** in a new tab. Give the token a name in the Token Name field, check the box to grant it API scope, and click 'Create personal access token'. This will insert an element (that contains the token) into the page just above the section where you entered the details.

Copy this token and head back over to the project, then select Settings>CI/CD and under the Variables section click the Add variable button. Paste the token into the Value field, set the key to `SETTINGS__GITLAB_ACCESS_TOKEN` and ensure that both the 'Protect variable' and 'Mask variable' boxes are checked. Then press the 'Add variable' button.

These checkboxes tell GitLab to redact the variable if it's detected in CI/CD logs and to only provide it when

» THE PROS AND CONS OF OWASP'S DEPENDENCYTRACK

If you have the time and resources to manage and maintain it then it's possible to build a complete self-hosted SCA solution, including all of the reporting features that you need ,by utilising The OWASP's Foundation DependencyTrack. This uses something called a Software Bill of Materials (SBOM) to carry out a lot of the same analytics and issue tracking as *Snyk*. SBOM of course being the output of another OWASP project: CycloneDX. DependencyTrack has a lot of good things going for it: a modern "dark mode" UI, policy compliance testing, impact analysis, time-series metrics, various SSO solutions, and it's completely open source, too.

However, it also has some significant downsides, such as needing to use a multitude of different tools to build the SBOM (which makes it difficult for projects that use multiple languages). It's also another service that you need to host somewhere and maintain (meaning yet more load on the DevOps Team).

In principle it's a great project, but in practice it's like a lot of OWASP projects; brilliant, but difficult and despite being free, it's very costly to implement as an individual or small company.

the CI/CD Pipeline is running on a protected branch. This is important because failing to protect this variable like this would mean that anyone who can push to your project (such as external contributors) could get a hold of your token, and use it to do whatever they wanted through the GitLab API as if they were you.

Finally, you'll need to update the **.gitlab-ci.yml**. To the `stages` section add the line `- composition-analysis` and then below that you'll need to add the **.dependabot-gitlab** template, which is as follows:

```
.dependabot-gitlab:
  stage: composition-analysis
  image:
    name: docker.io/andrcuns/dependabot-gitlab:0.4.4
    entrypoint: [""]
  variables:
    GIT_STRATEGY: none
    RAILS_ENV: production
    SETTINGS__STANDALONE: "true"
    SETTINGS__GITLAB_URL: $CI_SERVER_URL
  only:
    - main
    - merge_requests
  before_script:
    - cd /home/dependabot/app
  script:
    - bundle exec rake "dependabot:update[$CI_
PROJECT_PATH,$PACKAGE_
MANAGER,$DIRECTORY]"
```

Then add the `dependabot-pip` job as follows:

```
dependabot-pip:
  extends: .dependabot-gitlab
  variables:
    PACKAGE_MANAGER: pip
    DIRECTORY: /src
```

When you're all done the file should look something like this:

```
stages:
- static-analysis
- composition-analysis
.dependabot-gitlab:
  stage: composition-analysis
  image:
    name: docker.io/andrcuns/dependabot-gitlab:0.4.4
    entrypoint: [""]
  variables:
    GIT_STRATEGY: none
    RAILS_ENV: production
    SETTINGS__STANDALONE: "true"
    SETTINGS__GITLAB_URL: $CI_SERVER_URL
  before_script:
    - cd /home/dependabot/app
  script:
    - bundle exec rake "dependabot:update[$CI_
PROJECT_PATH,$PACKAGE_
MANAGER,$DIRECTORY]"
dependabot-pip:
  extends: .dependabot-gitlab
  variables:
    PACKAGE_MANAGER: pip
    DIRECTORY: /src
.
.
.
```

(Where the ... is the static analysis jobs from the previous tutorial)

Finally, `git add` the **.gitlab-ci.yml** and the **.gitlab/dependabot.yml** files, then `git commit -m "Added Dependabot SCA stage"` and `git push` the changes up GitLab.

What's interesting to note here is that the variables in the **.gitlab-ci.yml** file that point to the same values as the ones in the **dependabot.yml** are selectors for that configuration. This means that if you wanted to extend this to support scanning a Docker Container you would need to add a directive to the **dependabot.yml** like in the following code:

```
  - package-ecosystem: "docker"
    directory: "/"
    schedule:
      interval: "daily"
```

and a selector to the **gitlab-ci.yml**:

```
dependabot-docker:
  extends: .dependabot-gitlab
  variables:
    PACKAGE_MANAGER: docker
    DIRECTORY: /
```

which would then ensure that your docker files are kept up to date with the latest security patches.

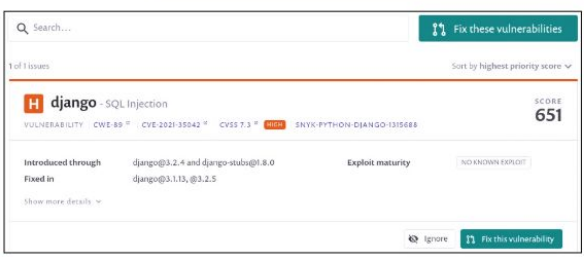
Dependabot or depend-on-Snyk?

Out of the box, this *Dependabot* implementation is going to check for dependencies with known vulnerabilities on every commit to the 'main' branch and any commit to a branch referenced in a Merge Request. If there's a patch required and there isn't an open Merge Request created by it to fix the issue, then it'll create Merge Requests containing the minimum possible change.

It's also possible to configure a scheduled pipeline run to trigger the *Dependabot* scan jobs periodically. This means that, just like *Snyk*, Merge Requests will still be opened to keep you up to date even if you're not actively working on the project at the time.

No doubt as the original *Dependabot* gets more integrated with GitHub, this will diverge from the GitLab fork, which is likely to remain an outside project with low integration despite the maintainer now working at GitLab. This is because GitLab has been working on its own fully integrated paid solution, *Gemnasium*, since 2018. Unfortunately however, at the time of writing GitLab doesn't offer a free tier of *Gemnasium*, so if you wanted to use it for your open-source projects then you need to build everything from source and set up a similar pipeline to the *Dependabot* one that's shown in this article.

Signs of the *Dependabot* divergence are already starting to be visible, with the GitHub integration now



Snyk's Vulnerability view gets straight to the point and shows you everything you need to know.

Snyk maintains a history of dependencies and vulnerabilities so that you can quickly see when a vulnerability was discovered and when you were exposed.

being a single click operation and the inline reports in its Pull Requests providing information about the vulnerability along with a "Compatability score". GitHub's version of *Dependabot* and the tooling being built around it is becoming a significant threat to GitLab's claim of being the "leading integrated product for the entire DevOps lifecycle". GitLab's failure to offer free access to the whole stack for open source projects could cost them a lot of marketing power.

It will be interesting to see whether or not GitLab will allow the continued work on the *Dependabot* fork since hiring its maintainer. Will this become a viable option for Open Source projects that want to stay out of Microsoft's ecosystem, or is it going to wither away?

DependaNOT!

While the net result is the same for both of the solutions covered in this tutorial (a new PR created automatically that upgrades the Django version) - *Dependabot* (both this open source GitLab version and the GitHub version) lacks the full feature set and depth of user experience offered by *Snyk*. Key features such as reporting, licence checking, and issue tracking that are found in *Snyk* (WhiteSource, and to an extent GitLab's "Ultimate" package), are currently not available in the GitLab version of *Dependabot* and are still not up to a competitive level in the GitHub version.

If you're working in the financial sector (or anywhere that handles payment card details for that matter), then it's a no-brainer. You're actually required to have the reporting capabilities and maintain a "vulnerability management program" in order to comply with card payment industry standards such as PCI-DSS. So getting hold of a pre-built compliant solution will save you a lot of time and money. Snyk's offering really makes a lot of sense when you consider the time it takes to set up, the features it provides, and their support for the Open Source community. Not to mention they have some fantastic plugins for popular IDEs to help you prevent problems from occurring early on in the development life-cycle.

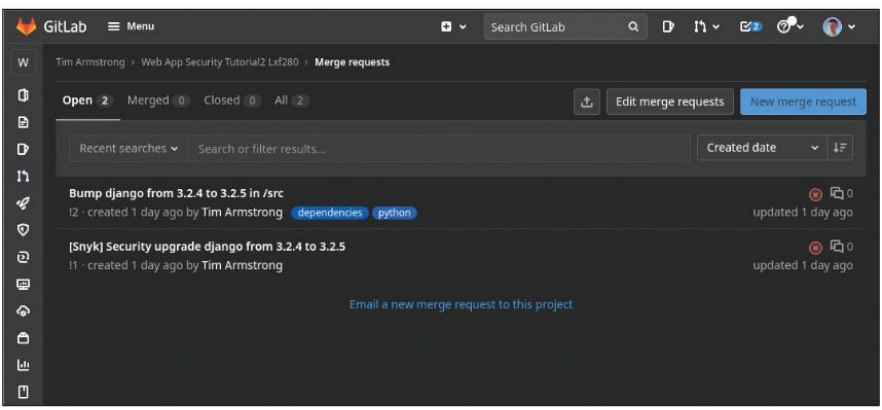
However, if you're working in a small company that doesn't directly handle customer-centric elements such as credit cards or personally identifiable information, or are busy developing some kind of super-secret project that doesn't need compliance reporting, then you can probably get by with something like the *Dependabot* solution provided in this tutorial.

Setting up dedicated accounts for bots like *Snyk* or *Dependabot* makes auditing changes easier. Because you know that the bot should only ever change specific files, it becomes easy to identify malicious activity. LXF

» DATA SECURITY STANDARDS

A lot of engineers like to rail against standards as pointless box-checking and in some cases they are. Especially when it comes to things like PCI-DSS. Fundamentally, any engineer can look at PCI-DSS and say honestly that it's obvious, and that all of these things should be implemented anyway. But when you ask them if they do, the answer is most often, "No, not all of it". When you look at certain companies that are compliant (or are applying for certification) and see how much effort they take to "limit the impact" of complying, you start to ask yourself, "If this is all obvious BCoP, then why is everyone afraid of it?".

These standards are here to provide a benchmark that consumers can look at and trust that the company complies to the bare minimum of what it should be in this day and age. Even self-certifying something like ISO 27001 means that you consider your user's data worthy of protection. These aren't high-bars to pass, so any company that asks for your data and doesn't show a compliance notice anywhere is not worthy of your business, and at some point, you can bet you'll be reading about them in the news.



When all is said and done, Dependabot and Snyk resulted in identical merge requests patching the vulnerability.

PYTHON

Credit: https://inventwithpython.com/pygame/chapter4.html

Sliding puzzles with Pygame and Python

Flex your coding muscles with **Calvin Robinson**, who reveals how to create a retro sliding puzzle game in Python.

OUR EXPERT

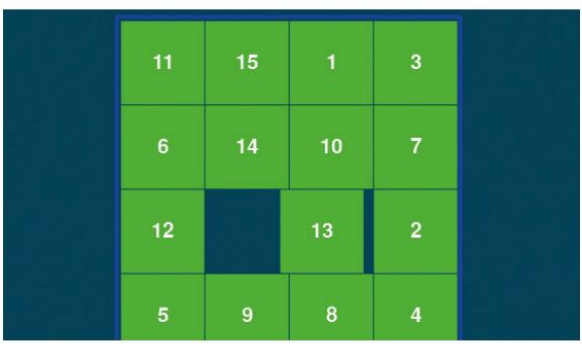
Calvin Robinson is a former assistant principal and computer science teacher. He has a degree in computer game design and coding.

For this game-coding outing we're taking a look at implementing a classic sliding tile puzzle game, also known as the sliding block game. Usually a two dimensional video game based on a retro board game where tiles are placed on a block in a mixed fashion, the player has to sort the tiles into an order. This may be words, pictures or numbers. The board contains one blank space, enabling the player to move one tile at a time. Despite looking so simple, they can take quite a while to complete.

We're looking at the source code by Al Sweigart reproduced with permission and available from **https://inventwithpython.com/pygame/chapter4.html** this version of the sliding puzzle consists of 16 spaces in a four by four grid, including one blank and 15 sequential numbers; it's probably the most common variant of sliding tiles games. Our player will have the ability to click tiles, and provided there's an empty space next to said tile, it will shift up, down, left or right into the empty space. At the end of the game, in order to win, the player must have one to 15 tiles sorted in order.

We'll need to make sure we have Python3 and *Pygame* installed. If you're running a Debian-based distro, launch a terminal window and type `sudo apt-get install python3 pip3` . Then when it's installed run `pip3 install pygame` and you'll be good to go. With that done, you'll then need to either launch the Python IDLE editor, or your favourite text-based editor. If using a text editor, save the new document as **slidepuzzle.py**; it can then be run from the terminal with `python3 slidepuzzle.py` provided you're in the right directory.

If you prefer to use Python's inbuilt IDLE, remember to press File>New File before you begin coding – so many times this author has witnessed people typing directly into the Shell window instead of creating a new Edit window. You can only save and edit your code if it's been written in the Edit window. Shell is for on-the-fly coding only. You can tell Python to always launch an Edit window on launch, instead of defaulting with a Shell window, by going to Options>Settings>General, and ticking Open Edit Window next to At Startup. To save and test your game at any point, press F5.



It's difficult to capture a 2D sliding animation in a screenshot!

No need to reinvent the wheel

Pygame is a fantastic resources for these types of projects; the *Pygame* module offers a host of classes we can tap into to do seemingly simple things like drawing shapes on-screen, without having to get into the nitty gritty of vector graphics. So, let's start off by importing the *Pygame* libraries we need:

```
import pygame, sys, random
from pygame.locals import *
```

We've got quite a number of constants to declare and initialise at the start, see the code for the full list:

```
BOARDWIDTH = 4
BOARDHEIGHT = 4
TILESIZE = 80
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
BLACK =      ( 0,  0,  0)
WHITE =      (255, 255, 255)
BRIGHTBLUE =   ( 0,  50, 255)
DARKTURQUOISE = ( 3,  54,  73)
GREEN =      ( 0, 204,  0)
BGCOLOR = DARKTURQUOISE
TILECOLOR = GREEN
TEXTCOLOR = WHITE
BORDERCOLOR = BRIGHTBLUE
XMARGIN = int((WINDOWWIDTH - (TILESIZE *
BOARDWIDTH + (BOARDWIDTH - 1))) / 2)
```

```
YMARGIN = int((WINDOWHEIGHT - (TILESIZE * BOARDHEIGHT + (BOARDHEIGHT - 1))) / 2)
```

We've set the board width and height to a 4x4 grid, and the screen resolution at 640x480. The framerate is capped at 30fps. We've also defined a number of colours (white, bright blue, dark turquoise and green) using RGB values. RGB count the amount of red, green and blue light in a pixel, with 255 being full and 0 being none. Therefore, 0,0,0 would be black and 255,255,255 would be white, displaying the full spectrum of colour. We've also set our font size and the directional values.

We'll need a `main()` function to create the handling of our game window, clock/timer, fonts and more. We'll declare some local variables for these elements:

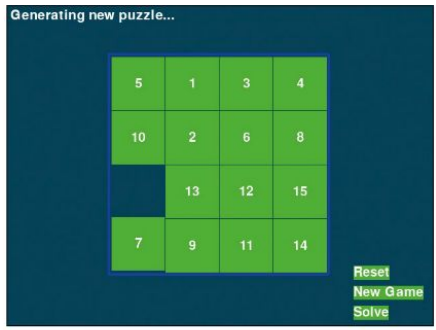
```
def main():
  global FPSCLOCK, DISPLAYSURF, BASICFONT, RESET_SURF, RESET_RECT, NEW_SURF, NEW_RECT, SOLVE_SURF, SOLVE_RECT
  pygame.init()
  FPSCLOCK = pygame.time.Clock()
  DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
  pygame.display.set_caption('Slide Puzzle')
  BASICFONT = pygame.font.Font('freesansbold.ttf', BASICFONTSIZE)
  RESET_SURF, RESET_RECT = makeText('Reset', TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 120, WINDOWHEIGHT - 90)
  NEW_SURF, NEW_RECT = makeText('New Game', TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 120, WINDOWHEIGHT - 60)
  SOLVE_SURF, SOLVE_RECT = makeText('Solve', TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 120, WINDOWHEIGHT - 30)
  mainBoard, solutionSeq = generateNewPuzzle(80)
  SOLVEDBOARD = getStartingBoard()
  allMoves = []
```

And now to set up the main game loop with our start message and our end game message, as well as instructions to draw the board itself and the direction tiles should slide into:

```
  while True:
    slideTo = None
    msg = 'Click tile or press arrow keys to slide.'
    if mainBoard == SOLVEDBOARD:
      msg = 'Solved!'
    drawBoard(mainBoard, msg)
    checkForQuit()
```

Continue this loop with some conditional IF statements, with an event handling loop:

```
    for event in pygame.event.get():
      if event.type == MOUSEBUTTONUP:
        spotx, spoty = getSpotClicked(mainBoard, event.pos[0], event.pos[1])
        if (spotx, spoty) == (None, None):
          if RESET_RECT.collidepoint(event.pos):
            resetAnimation(mainBoard, allMoves)
            allMoves = []
          elif NEW_RECT.collidepoint(event.pos):
            mainBoard, solutionSeq = generateNewPuzzle(80)
            allMoves = []
          elif SOLVE_RECT.collidepoint(event.pos):
            resetAnimation(mainBoard, solutionSeq + allMoves)
```



Here's the game generating tiles in readiness for a new session.

```
          allMoves = []
        else:
          blankx, blanky = getBlankPosition(mainBoard)
          if spotx == blankx + 1 and spoty == blanky:
            slideTo = LEFT
          elif spotx == blankx - 1 and spoty == blanky:
            slideTo = RIGHT
          elif spotx == blankx and spoty == blanky + 1:
            slideTo = UP
          elif spotx == blankx and spoty == blanky - 1:
            slideTo = DOWN
      elif event.type == KEYUP:
        if event.key in (K_LEFT, K_a) and isValidMove(mainBoard, LEFT):
          slideTo = LEFT
        elif event.key in (K_RIGHT, K_d) and isValidMove(mainBoard, RIGHT):
          slideTo = RIGHT
        elif event.key in (K_UP, K_w) and isValidMove(mainBoard, UP):
          slideTo = UP
        elif event.key in (K_DOWN, K_s) and isValidMove(mainBoard, DOWN):
          slideTo = DOWN
```

We're checking if the player has clicked one of the option buttons for a new game, to reset the game, or to solve the game. We're also checking if the tile is next to a black space. To solve the puzzle, we're not using super smart artificial intelligence. Instead, we're making our program take note of every tile it creates when inserting them at random, and then using that pattern in reverse to solve the puzzle. Every up becomes a down, and every left move becomes a right, until eventually the board is back in sequential order. We use our array `allMoves = []` to store these values.

Afterwards, we're checking to see if the player has clicked a tile to move it up, down, left or right. Following that, we'll need to slide said tile, finish our loop with the following code:

```
    if slideTo:
      slideAnimation(mainBoard, slideTo, 'Click tile or press arrow keys to slide.', 8)
      makeMove(mainBoard, slideTo)
      allMoves.append(slideTo)
    pygame.display.update()
    FPSCLOCK.tick(FPS)
```

This will show the slide animation and take note of any slides. We then update our window, in typical *Pygame* fashion.

QUICK TIP

This tutorial is based on the source code by Al Sweigart, available from https://invent withpython.com/pygame/chapter4.html.

We'll want a way for our players to quit the game easily and efficiently. Python can get a little frustrated if a quit command isn't passed through *Pygame*:

```
def terminate():
  pygame.quit()
  sys.exit()
def checkForQuit():
  for event in pygame.event.get(QUIT):
    terminate()
  for event in pygame.event.get(KEYUP):
    if event.key == K_ESCAPE:
      terminate()
    pygame.event.post(event)
```

With these two functions we're checking for an escape or exit command and quitting our game.

On the other end of the spectrum, we'll need a command to start a new game:

```
def getStartingBoard():
  counter = 1
  board = []
  for x in range(BOARDWIDTH):
    column = []
    for y in range(BOARDHEIGHT):
      column.append(counter)
      counter += BOARDWIDTH
    board.append(column)
    counter -= BOARDWIDTH * (BOARDHEIGHT - 1) +
BOARDWIDTH - 1
  board[BOARDWIDTH-1][BOARDHEIGHT-1] =
BLANK
  return board
def getBlankPosition(board):
  for x in range(BOARDWIDTH):
    for y in range(BOARDHEIGHT):
      if board[x][y] == BLANK:
        return (x, y)
```

This looks slightly more complicated than our other functions so far, but essentially it's setting up the board with the correct height and width. `getBlankPosition` will store the location of our blank space. This means that if our player clicks a tile next to the blank space, that tile will be able to move into it with the `slideTo` method. To move our tiles we'll need a function – we'll just show the UP and DOWN conditions here:

```
def makeMove(board, move):
  blankx, blanky = getBlankPosition(board)
  if move == UP:
    board[blankx][blanky], board[blankx][blanky + 1] =
board[blankx][blanky + 1], board[blankx][blanky]
  elif move == DOWN:
    board[blankx][blanky], board[blankx][blanky - 1] =
board[blankx][blanky - 1], board[blankx][blanky]
```

Before we use the `makeMove` function, it's important that we check if the move is valid.

```
def isValidMove(board, move):
  blankx, blanky = getBlankPosition(board)
  return (move == UP and blanky != len(board[0]) - 1) or \
      (move == DOWN and blanky != 0) or \
      (move == LEFT and blankx != len(board) - 1) or \
      (move == RIGHT and blankx != 0)
```

To set up the board with random tiles, we'll need to create a `getRandomMove` function.

```
def getRandomMove(board, lastMove=None):
  validMoves = [UP, DOWN, LEFT, RIGHT]
  if lastMove == UP or not isValidMove(board, DOWN):
    validMoves.remove(DOWN)
```

and repeat for additional directions:

```
  return random.choice(validMoves)
```

Instead of just using `random.choice()`, we've developed our own method so that we don't end up with the same random move two times in a row, or worse, moving a tile in one direction and then randomly moving it back in the opposite direction. Our method makes these considerations.

On the grid

When we're happy moving individual tiles around, we'll need to make sure that we do so within the grid. Let's start by creating a function to convert pixels into grid coordinates:

```
def getLeftTopOfTile(tileX, tileY):
  left = XMARGIN + (tileX * TILESIZE) + (tileX - 1)
  top = YMARGIN + (tileY * TILESIZE) + (tileY - 1)
  return (left, top)
```

We'll need to do a similar thing with clicks, in assigning clicks to the tile connected to the pixel that was actually clicked:

```
def getSpotClicked(board, x, y):
    for tileX in range(len(board)):
    for tileY in range(len(board[0])):
      left, top = getLeftTopOfTile(tileX, tileY)
      tileRect = pygame.Rect(left, top, TILESIZE,
TILESIZE)
      if tileRect.collidepoint(x, y):
        return (tileX, tileY)
  return (None, None)
```

Let's draw our tiles, text and the actual board:

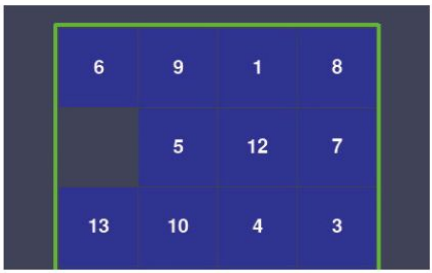
```
def drawTile(tilex, tiley, number, adjx=0, adjy=0):
  left, top = getLeftTopOfTile(tilex, tiley)
  pygame.draw.rect(DISPLAYSURF, TILECOLOR, (left +
adjx, top + adjy, TILESIZE, TILESIZE))
  textSurf = BASICFONT.render(str(number), True,
TEXTCOLOR)
  textRect = textSurf.get_rect()
  textRect.center = left + int(TILESIZE / 2) + adjx, top +
int(TILESIZE / 2) + adjy
  DISPLAYSURF.blit(textSurf, textRect)
```

Each numbered tile will be drawn to our specification – colours and fonts were previously defined. Using

» MAKE IT HARDER!

To increase the difficulty of our sliding tiles game we could add timers. One method would be to count the number of seconds it takes to solve the puzzle and give the player an inverse number points. So, the quicker the solution, the more points earned. Those points could then be added on to a leader board with the player's name. Alternatively, display a countdown timer and give the player a set amount of time to solve the puzzle. Once the timer reaches zero, we could move some tiles around to make things more difficult, or we could entirely wipe the board and display a 'Game Over' message screen.

Instead of using timers to distribute points we could give the player points for the number of moves it takes them to complete the game. We'd only have to add a timer array into our game and count the number of clicks, then at the end we could give the player an inverse points, so that the fewer amount of clicks it took them to complete the puzzle, the more points they earn. We could then display those points of the screen, and again, potentially develop a leader board.



Our game with an altered colour palette.

Pygame we'll create a surface and a rectangle to display text on the screen:

```
def makeText(text, color, bgcolor, top, left):
  textSurf = BASICFONT.render(text, True, color,
bgcolor)
  textRect = textSurf.get_rect()
  textRect.topleft = (top, left)
  return (textSurf, textRect)
```

New boards are painted over anything displayed:

```
def drawBoard(board, message):
  DISPLAYSURF.fill(BGCOLOR)
  if message:
    textSurf, textRect = makeText(message,
MESSAGECOLOR, BGCOLOR, 5, 5)
    DISPLAYSURF.blit(textSurf, textRect)
  for tilex in range(len(board)):
    for tiley in range(len(board[0])):
      if board[tilex][tiley]:
        drawTile(tilex, tiley, board[tilex][tiley])
  left, top = getLeftTopOfTile(0, 0)
  width = BOARDWIDTH * TILESIZE
  height = BOARDHEIGHT * TILESIZE
  pygame.draw.rect(DISPLAYSURF, BORDERCOLOR,
(left - 5, top - 5, width + 11, height + 11), 4)
  DISPLAYSURF.blit(RESET_SURF, RESET_RECT)
  DISPLAYSURF.blit(NEW_SURF, NEW_RECT)
  DISPLAYSURF.blit(SOLVE_SURF, SOLVE_RECT)
```

Using `getBlankPosition()` we can slide our tiles into the blank space with a nice little animation. This snippet only shows the code for dealing with UP and DOWN:

```
def slideAnimation(board, direction, message,
animationSpeed):
  blankx, blanky = getBlankPosition(board)
  if direction == UP:
    movex = blankx
    movey = blanky + 1
  elif direction == DOWN:
    movex = blankx
    movey = blanky - 1
```

and repeat for additional directions:

```
  drawBoard(board, message)
  baseSurf = DISPLAYSURF.copy()
  moveLeft, moveTop = getLeftTopOfTile(movex, movey)
  pygame.draw.rect(baseSurf, BGCOLOR, (moveLeft,
moveTop, TILESIZE, TILESIZE))
  for i in range(0, TILESIZE, animationSpeed):
    checkForQuit()
    DISPLAYSURF.blit(baseSurf, (0, 0))
    if direction == UP:
      drawTile(movex, movey, board[movex][movey], 0, -i)
    if direction == DOWN:
      drawTile(movex, movey, board[movex][movey], 0, i)
```

and repeat for additional directions:

```
    pygame.display.update()
    FPSCLOCK.tick(FPS)
```

Again, we're not checking if the move is valid here because we have a function for that already. We simply prepare the `baseSurface` and draw a black space over the moving tile. We'll need a way to recreate the puzzle or reset the game:

```
def generateNewPuzzle(numSlides):
  sequence = []
  board = getStartingBoard()
  drawBoard(board, '')
  pygame.display.update()
  pygame.time.wait(500)
  lastMove = None
  for i in range(numSlides):
    move = getRandomMove(board, lastMove)
    slideAnimation(board, move, 'Generating new
puzzle...', animationSpeed=int(TILESIZE / 3))
    makeMove(board, move)
    sequence.append(move)
    lastMove = move
  return (board, sequence)
```

Here we're making `numSlides` the number of moves and then animating them. We'll need a way to reset our animations. To do this, we'll reverse `allMoves`:

```
def resetAnimation(board, allMoves):
  revAllMoves = allMoves[:]
  revAllMoves.reverse()
  for move in revAllMoves:
    if move == UP:
      oppositeMove = DOWN
```

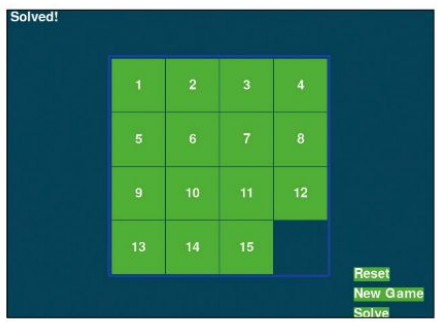
and repeat for additional directions:

```
    slideAnimation(board, oppositeMove, '',
animationSpeed=int(TILESIZE / 2))
    makeMove(board, oppositeMove)
```

To wrap up, instruct our program to run our `main()` function on launch, and we're good to go!

```
if __name__ == '__main__':
  main()
```

Hit F5 if you're using Python IDLE, or **python3 slidepuzzle.py** in terminal, otherwise, and you should now have a fully functional slide tile puzzle game!



Fin! Now, do you want to play another game?

QUICK TIP

Our colour scheme is pretty basic, try improving the look and feel of the game by altering the colour arrays. Remember, these are constants we declared early on in the code with RGB (Red, Green, Blue) values.

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 LXFDVD

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.

ONCE YOU POP!_...
MIN SPECS: 64-BIT CPU, 2GB RAM, 16GB DISK

Pop!_OS 21.04

64-bit

We might not be sure about the punctuation in Pop!_OS. But we are certain that since its inception in 2017 it's fast become one of our favourite distros. Developed by System76 and pre-installed on its finely crafted Linux laptops and desktops, it's great for beginners and pros alike.

Pop has been innovating the bejesus out of the Gnome desktop lately. Most notably, the previous Pop LTS release (June 2020) introduced a tiling window mode, encouraging a smoother, keyboard-driven workflow. That evolved in the interim 20.10 release (with stackable tiles and exceptional floating windows), and in this fresh outing it's better than ever. The desktop at large is now known as COSMIC (a beautifully wrought acronym for Computer Operating System Main Interface Components) and as its centrepiece now includes a custom dock for launching and switching applications. This will be great news if you find Gnome's Activity View cumbersome to work with.

There's a new, minimalist launcher too. Just hit the Super (Windows) key and type a few characters to search applications (and switch between open ones). The launcher can be expanded with plugins, so in theory you can have it search your favourite websites. System76 has put a lot of thought into their keyboard shortcuts, which occasionally deviate from how things work in vanilla Gnome. Fortunately, help is always at hand, with visible shortcut hints, as well as a cheatsheet for the auto-tiling mode.

For laptop users, this release also goes big on touchpad gestures. In particular a four-fingered swipe left or right will now summon the Workspaces and Applications views respectively. And up or down will switch between workspaces. Swiping with three fingers will switch between windows. Speaking of workspaces, Pop includes a shortcut to these in the top-left that summons a non-intrusive, transparent switcher. This does shunt the Activities shortcut along, which might break your workflow. We found our desktop much cleaner after we disabled both. Windows also have minimise buttons by default, which makes more sense in a tiling environment.

If this brave new way of working (or creating or watching cat videos) seems tricky, then don't worry. Pop remains very customisable. On first boot the Welcome application offers three dock arrangements: no dock (for Gnome traditionalists); a full length dock; or a floating macOS type of affair. So you can make it as traditional or avant-garde as you like.

Pop eschews Ubuntu's Snaps, but fear not – the latest software is available as Flatpaks from the Pop!_Shop.

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.

ARTEFACTS AND RELICS

MIN SPECS: A REALLY OLD PC

Historic distros

32-bit

The Linux kernel was released score and 10 years ago, and to acknowledge this achievement we've compiled a selection of our favourite distros from yesteryear. These can't be booted from the DVD, but have a look in the **olddistros/** folder on the disc and try booting the ISOs in a virtual machine.

We really wouldn't recommend using these aging distributions on real hardware, especially if it's connected to the internet. What's more, they're unlikely to work on hardware that wasn't invented at the time of their release, but of course we can't stop you from trying. None of these distros support booting from a USB stick, but you never know, it might be possible with *Unetbootin* or such.

In tandem with Les Pounder's historic distros feature (*see page 68*) we've got a Debian 2.0 (from 1998) QEMU image. Codenamed "Hamm", this release had a mighty 1,500 packages available in the repos. The *apt* command line utility wouldn't be invented until Debian 3.0, so all package management is done through `dpkg`. So you're free to recreate the quintessentially late-90s dependency hell experience.

Next, jumping forward five years, we have Klaus Knopper's Knoppix. Knoppix was one of the first distros able to boot directly to a graphical live environment. This version includes some of the in-vogue software of the era, including KDE 3, *GIMP 1.2*, *xmms* and *OpenOffice*. A whole 96MB of RAM was needed to use the GUI, which even then was low. This edition defaults to Knopper's native German, but you can change this in the Land und Sprache settings. While you're at it, you'll probably want to change the keyboard layout too. Oh and if you want to change resolution you'll need to edit **XF86config**... the olden days!

The Nautilus file manager hasn't changed all that much in 15 years. If it ain't broke, don't... oh, hang on a minute.

Finally, we've got Ubuntu 6.06, which was the only release not to end with .04 or .10 due to a two-month delay. This was Ubuntu's first LTS release, which was jokingly said to stand for Late To Ship. According to contemporary accounts the wait was worth it though, as Dapper Drake shipped with improved power management capabilities, a new graphical installer (*Ubiquity*) and an optional tool called *NetworkManager* that promised to make life easier now that people were starting to roam around different networks. The Human theme made its inaugural appearance too, in all its orangey-brown glory.

Under the hood it's running kernel 2.6.15, so *Udev* (the kernel event interface that enables it to enumerate hardware dynamically) was still pretty new. But having even rudimentary hardware detection was a luxury in those halcyon days of the mid-2000s. And now, lest we drift off into nostalgic reverie, let us return to 2021 (*do we have to? – Ed*).

People were more willing to put up with crowded menus and ugly fonts in 2003.

NEW TO LINUX? START HERE...

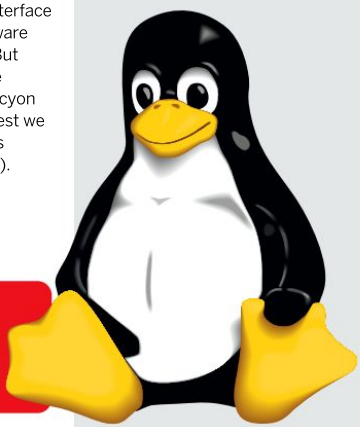
Never used a Linux before? Here are some handy resources:

- 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!

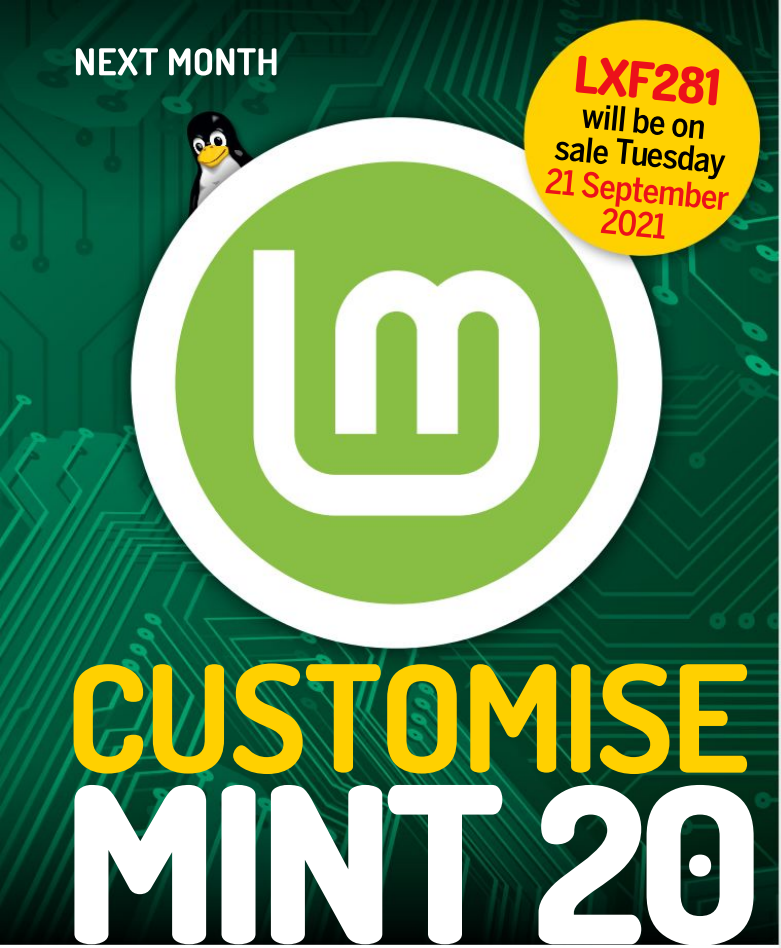
THE LXF LIBRARY

- **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

LXF281 will be on sale Tuesday 21 September 2021



CUSTOMISE MINT 20

Explore desktop options with the latest release of Mint 20.2 and get the best of all worlds.

Linux for creatives

We round up the best creative open source software and show you how Linux offers professional-level artistic tools.

Get your certificate

Encryption and online certificates are complex – discover how you can get your sites more easily signed and secured.

Cheaper, faster servers

We explore the world of Virtual Private Servers: why you'd want one, what to look for, what to avoid and how to get started.

Build a pin-hole Pi camera

Take one Pi, add the HQ camera and recreate the most basic of cameras in the most elaborate way possible. Because why not!?

Contents of future issues subject to change – as we might not survive another climate change summer :o(

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
Sickly editor Jonni Bidwell
Art editor Efrain Hernandez-Mendoza
Operations editor Cliff Hope
Group editor in chief Graham Barlow
Senior art editor Jo Gulliver
Editorial contributors
Tim Armstrong, Mats Tage Axelson, Neil Bothwick, Stuart Burns, Christian Cawley, Matthew Hanson, Matthew Holder, Jon Masters, Nick Peers, Les Pounder, Calvin Robinson, Mayank Sharma, Shashank Sharma, Mihalis Tsoukalous, Alexander Tolstoy
Cover illustration magictorch.com
Raspberry Pi is a trademark of the Raspberry Pi Foundation. Tux credit: Larry Ewing (lewing@isc.tamu.edu) and *The GIMP*.

ADVERTISING
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